



**An Efficient Resource Allocation in Grid Computing Environment**

A Dissertation

Submitted By

**Baljit Kaur**

**11000553**

To

**Department of Computer Science and Engineering**

In partial fulfilment of the Requirement for the

Award of the Degree of

**Master of Technology in Computer Science and Engineering**

Under the guidance of

**Harsh Bansal**

**Assistant Professor**

(July 2015)

# PAC APPROVAL PAGE



School of: Computer Science and Engineering

## DISSERTATION TOPIC APPROVAL PERFORMANCE

Name of the student : BALJIT KAUR Registration No : 11000553  
Batch : 2010-2015 Roll No : RK2005A10  
Session : 2014-2015 Parent Section : K2005

Details of Supervisor:  
Name : HARSH BANSAL Designation : Assistant Professor  
UID : 16866 Qualification : M.TECH  
Research Exp. : 2 years

Specialization Area: Database (pick from list of provided specialization areas by DAA)

Proposed Topics:-

1. AN EFFICIENT AGENTLESS RESOURCE ALLOCATION IN GRID COMPUTING ENVIRONMENT.
2. USER BASED EFFICIENT RESOURCE ALLOCATION IN GRID COMPUTING ENVIRONMENT.
3. DYNAMIC GRAVITATIONAL BASED RESOURCE ALLOCATION IN GRID COMPUTING.

*Harsh - 16866*  
Signature of supervisor

PAC Remarks:

APPROVAL OF PAC CHAIRMAN

Signature:

Date:

\*Supervision should finally encircle one topic out of three proposed topics and put up for an approval before Project Approval Committee (PAC).

\*Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation final report.

\*One copy to be submitted to supervisor.

## **CERTIFICATE**

This is to certify that Baljit Kaur has completed M. Tech. dissertation titled AN EFFICIENT RESOURCE ALLOCATION IN GRID COMPUTING ENVIRONMENT under my guidance and supervision. To the best of my knowledge, the present work is the result of her original investigation and study. No part of the dissertation has ever been submitted for any other degree or diploma.

The dissertation is fit for the submission and the partial fulfilment of the conditions for the award of M. Tech. Computer Science and Engineering.

Date:

**Signature of Advisor**

Name : Harsh Bansal

(Assistant Professor)

UID : 16866

Lovely Professional University

## ABSTRACT

Grid computing is distributed computing in which resources are distributed across various geographical locations. It aims at providing high computation power to the users to execute their applications by collaborating and integrating grid resources. Grid aims at utilizing distributed idle nodes so as to enhance performance, resource sharing, increase availability and extensibility. Task scheduling and resource allocation become more complex with the ever increase in grid size. The main problem which we face while scheduling the job(s) in grid environment is its dynamicity. The resources are dynamic in nature means they can join or leave at any time. Therefore it is difficult to manage those resources. Here we are going to propose the mechanism to manage them well by using an efficient algorithm to assign job(s) to grid resource(s) efficiently. In this mechanism, user based scheduling will be done instead of job based scheduling to allocate the resource(s) to the job(s). The proposed algorithm categorizes the resources into high, medium, and low end resources based on their configuration. In this way it helps to reduce the search time for the fittest available resource(s) on the basis of SLA type of the user(s) than the existing technique like AHSWDG. This technique needs to find the fittest resources from all the available resources. The proposed technique activates the services to provide fault tolerance according to priority of user to provide the reliability and optimum solution. The existing technique such as FIFO and AHSWDG can't handle any failure, failed job need to resubmit to the Grid Data Center. The job success rate of the proposed technique is more than FIFO and AHSWDG. The proposed technique aims at providing optimal and reliable solution, reducing search time and failure rate, and to efficiently allocating the resources to job(s) based on the user priority.

## **ACKNOWLEDGEMENT**

Foremost, I would like to express my gratitude to my advisor Mr. Harsh Bansal for continuous support in my research, for his patience, motivation and encouragement. Without his wise counsel, it would have been impossible to complete the dissertation report in such a way.

I would also like to thank other faculty members of CSE/IT department of Lovely Professional University for their intellectual support throughout the course of this work.

I am grateful to my friends who helped me a lot in searching for the information related to research work. I also owe my gratitude to my parents who provided me with the resources that helped me in the completion of this research work.

**Name:** Baljit Kaur,

**Reg. No.:** 11000553.

## DECLARATION

I hereby declare that the dissertation entitled, **An Efficient Resource Allocation in Grid Computing Environment**, submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date:

Investigator

Reg. No.: 11000553

# TABLE OF CONTENTS

1	INTRODUCTION .....	1
1.1	Grid Resource.....	1
1.1.1	Types of Grid resources .....	1
1.2	Resource Allocation .....	1
1.2.1	Resource Scheduling.....	2
1.2.2	Code Transfer.....	3
1.2.3	Data Transmission .....	3
1.2.4	Monitoring .....	3
1.3	Resource Management .....	3
1.3.1	Centralized organization .....	4
1.3.2	Hierarchical organization.....	4
1.3.3	Decentralized organization .....	4
1.3.4	Comparison of resource Management Organization .....	5
1.4	Types of grid .....	5
1.5	Resource Discovery Techniques .....	6
1.5.1	Query based resource discovery .....	6
1.5.2	Agent based resource discovery.....	6
2	REVIEW OF LITERATURE .....	7
3	PRESENT WORK .....	25
3.1	Scope of the study .....	25
3.2	Problem Formulation.....	25
3.3	Objectives:.....	27
3.4	Research Methodology.....	27
4	RESULTS AND DISCUSSION.....	32
4.1	Implementation.....	32
4.2	Experimental Results 1.....	32
4.2.1	Client Interface: - .....	32
4.2.2	Grid Master Module.....	34
4.2.3	Resource Joining Module .....	37
4.2.4	Resource Allocation Module .....	41
4.3	Experimental Results 2.....	42

4.3.1	Client.csv .....	42
4.3.2	Resources.csv .....	43
4.3.3	Jobtrace.csv .....	43
4.3.4	Resourceshigh.csv .....	44
4.3.5	Resourcesaverage.csv .....	45
4.3.6	Resourceslow.csv .....	45
4.4	Experimental Results 4.....	46
4.5	Experimental Results 5.....	53
4.5.1	Resource Utilization Comparison .....	53
4.5.2	Job Success Comparison.....	54
4.5.3	Job Failure Comparison .....	55
4.5.4	Search Time Comparison.....	56
4.5.5	Execution Time Comparison .....	58
4.5.6	Cost Comparison.....	59
5	CONCLUSION AND FUTURE WORK .....	61
6	LIST OF REFERENCES.....	62



## LIST OF FIGURES

Figure 1.1 Types of Grid Resources .....	2
Figure 1.2 Taxonomy of Resource Allocation Mechanism.....	3
Figure 1.3 Taxonomy of Resource Discovery Techniques.....	6
Figure 2.1 Types of Resource Discovery Techniques.....	10
Figure 2.2 Basic Grid Model.....	14
Figure 2.3 Distributed Clustering Algorithm.....	17
Figure 2.4 Heterogeneous Earliest Finish Time.....	18
Figure 2.5 Four different states of Grid Systems.....	19
Figure 2.6 Proposed SS-GA algorithm for Resource Allocation.....	23
Figure 3.1 Flow chart of AHSDWG.....	26
Figure 3.2 Flowchart of proposed algorithm.....	31
Figure 4.1 Netbeans IDE.....	32
Figure 4.2 Client Interface Window.....	33
Figure 4.3 Unsuccessful Login Output.....	33
Figure 4.4 Interface for Type A user.....	34
Figure 4.5 Output for Type A after successfully login.....	34
Figure 4.6 Interface for Type B user.....	35
Figure 4.7 Output for Type B after successfully login.....	35
Figure 4.8 Interface for Type C user.....	36
Figure 4.9 Output for Type C after successfully login.....	36
Figure 4.10 Interface for adding resources.....	37
Figure 4.11 Output after clicking ADD VM button.....	37

Figure 4.12 Interface after adding configuration of resource.....	38
Figure 4.13 Resource.csv file before adding new resource.....	39
Figure 4.14 Output after adding resource.....	39
Figure 4.15 Resource.csv file after adding new resource.....	40
Figure 4.16 Resourceaverage.csv file after adding new resource.....	40
Figure 4.17 Resource Allocation Interface.....	41
Figure 4.18 Client.csv file .....	42
Figure 4.19 Resource.csv file.....	43
Figure 4.20 JobTrace.csv file.....	44
Figure 4.21 Resourcehigh.csv file.....	44
Figure 4.22 Resourceaverage.csv file.....	45
Figure 4.23 Resourcelow.csv file.....	45
Figure 4.24 Output 1 for FIFO technique.....	46
Figure 4.25 Output2 for FIFO technique.....	47
Figure 4.26 Output for Type A user with FIFO technique.....	47
Figure 4.27 Output for Type B user with FIFO technique.....	48
Figure 4.28 Output for Type C user with FIFO technique.....	48
Figure 4.29 Output 1 for AHSWDG technique.....	49
Figure 4.30 Output 2 for AHSWDG technique.....	49
Figure 4.31 Output for Type A user with AHSWDG technique.....	49
Figure 4.32 Output for Type B user with AHSWDG technique.....	50
Figure 4.33 Output for Type C user with AHSWDG technique.....	50
Figure 4.34 Output 1 for EFFICIENT technique.....	51
Figure 4.35 Output 2 for EFFICIENT technique.....	52
Figure 4.36 Output for Type A user with EFFICIENT technique.....	52

Figure 4.37 Output for Type B user with EFFICIENT technique.....	52
Figure 4.38 Output for Type C user with EFFICIENT technique.....	53
Figure 4.39 Comparison graph of Resource Utilization.....	54
Figure 4.40 Comparison graph of Job Success.....	55
Figure 4.41 Comparison graph of Job Failure.....	56
Figure 4.42 Comparison graph of Search Time (in nanoseconds).....	58
Figure 4.43 Comparison graph of Execution Time (in millisecond).....	59
Figure 4.44 Comparison graph of Cost (in Rs).....	60

## LIST OF TABLES

Table 1.1 Comparison of Resource Management Organization.....	5
Table 4.1 Comparison Table of Resource Utilization.....	53
Table 4.2 Comparison Table of Job Success.....	55
Table 4.3 Comparison Table of Job Failure.....	56
Table 4.4 Comparison Table of Search Time.....	57
Table 4.5 Comparison Table of Execution Time.....	59
Table 4.6 Comparison Table of Cost.....	60

As scientific applications need large amount of resources for computational purpose that can't be provided by single workstation. The need of high and reliable computing power and simultaneous access to multiple distributed resources forces to focus on low cost and intelligent methodologies to share data and resources. Grid computing presents a solution for these types of problems or applications. Grid computing was introduced in 1990s. It provides a platform for virtual organizations to share their owned services. Grid computing is distributed computing which offers high performance by collaborating and integrating various resources like computing, communication, and storage distributed at different geographical locations. These resources are shared by resource-intensive user tasks to satisfy user requirements and to achieve high throughput. The development of various types of grid systems had inspired by these requirements.

### **1.1 Grid Resource**

Grid resource is an entity which is supposed to carry out an operation by an application. A resource can carry out one or more tasks based upon its capability.

#### **1.1.1 Types of Grid resources**

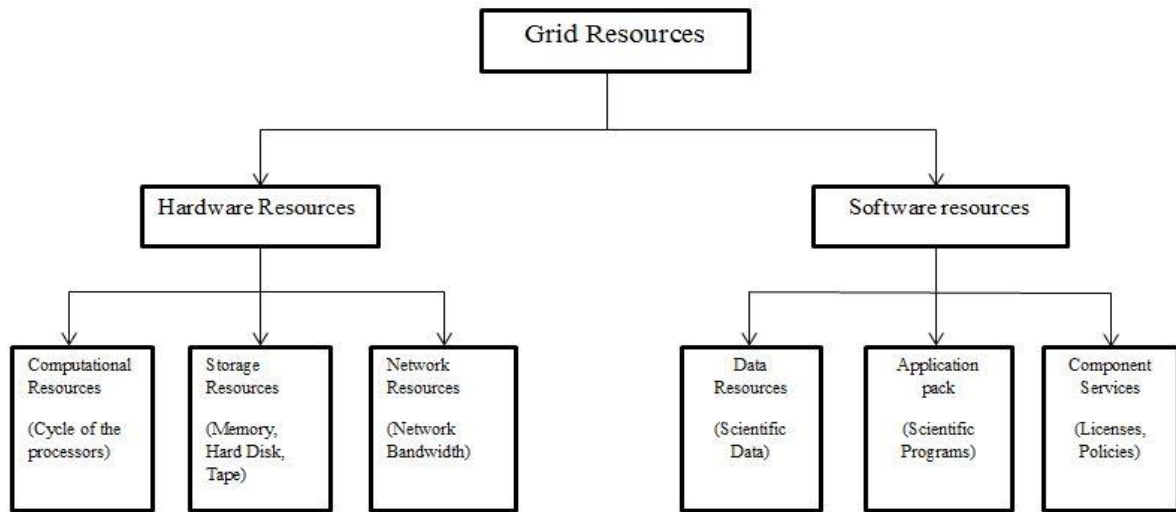
Grid resources include various types of computing, communication, and storage resources. These resources include computers, network bandwidth, storage space, sensors, software applications, data etc. The common types of grid resources are shown in Figure 1.1.

### **1.2 Resource Allocation**

As the size of grid technology keeps on increasing, resource allocation and scheduling has become more complex and challenging. This area has gained more attention of researchers from last few years. There are four main functions in grid resource allocation process:

- a) Resource Scheduling,
- b) Code Transfer,

- c) Data Transmission,
- d) Monitoring.



**Figure 1.1 Types of Grid Resources**

### 1.2.1 Resource Scheduling

In this process, applications are mapped to resources. An efficient resource is allocated to particular task so that it can perform the task in efficient manner to minimize its completion time. Hence improves the overall performance of grid. Three main phases of this process are as follows:

- a) Resource discovery ,
- b) Resource selection,
- c) Job execution.

#### 1.2.1.1 Resource Discovery

In this process, all the available resources are searched. Based on the result of this search, a list of available resources is generated.

#### 1.2.1.2 Resource Selection

In this process, best matched resource is selected from list that is generated in previous step. This matching is done on the basis of QoS criteria.

### 1.2.1.3 Job Execution

In this process, job is submitted to the best selected resource(s) for execution. The execution of job(s) is also monitored in this step.

### 1.2.2 Code Transfer

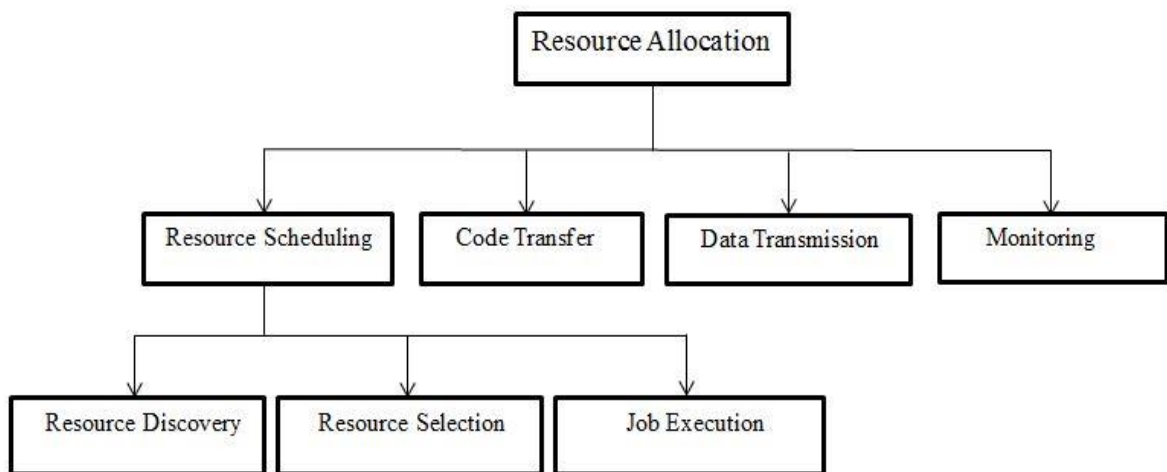
To execute each task, code of that task is needed by the resource. This process includes transfer of code of each task to the allocated resource so as to execute the task.

### 1.2.3 Data Transmission

In this process, whatever data is needed by the task that data is transferred for executing the task. When required data is transferred, then only execution of the task takes place.

### 1.2.4 Monitoring

In this step, availability, future reservations, usage, and capability of the resource are checked continuously. Monitoring is also defined as the process in which status information and characteristics of resources are collected. Resources are reserved in advance for the future use due to time availability of capable resources at specific time. The taxonomy of resource allocation mechanism is shown in Figure 1.2.



**Figure 1.2 Taxonomy of Resource Allocation Mechanism**

## 1.3 Resource Management

Resource Management is a process in which all the processes of resource allocation i.e. resource discovery, resource selection, resource scheduling, and system workloads are

managed. The process of authentication, accounting, authorization, fault tolerance is managed by Resource Management. The grid service which controls all the resource management processes is known as Resource Management System. Due to heterogeneity of grid resources, varying loads, dynamicity of resources, extensibility of grid, resource management (RM) becomes complex and challenging area.

To manage grid resources, various models are developed. Based upon the organization of components, Resource Management is of three types:

- a) Centralized organization,
- b) Hierarchical organization,
- c) Decentralized organization.

### **1.3.1 Centralized organization**

In this organization, there is one central server which manages the processes of resource management i.e. resource scheduling and allocation. The advantage of this organization is that it is easy to deploy. But there is no fault tolerance because central server is single point of failure and it also lacks scalability.

### **1.3.2 Hierarchical organization**

In this organization, resource managers are organized in tree like structure. There is one central manager and various low level schedulers. The central manager splits an application into various tasks and these tasks are further assign to low level schedulers. Now these low level schedulers further map these tasks to various grid resources. The central manager is responsible for the complete execution of an application. If the resources reside at same level, these resources can communicate directly without any need of intermediate node. The advantage of this organization is that it is more scalable and fault tolerable than centralized organization. But if central manager get fail then whole system fails. It also lacks site autonomy.

### **1.3.3 Decentralized organization**

In this organization, the managerial control is given to various nodes. Each managerial node can take its own independent decision. There is no any node which acts as central manager and having full-fledged information about the system. The advantage of this



organization is that it is more scalable, robust, and fault tolerable than both previously discussed organizations.

### 1.3.4 Comparison of resource Management Organization

The comparison of all three resource management organization is shown in Table 1.1

**Table 1.1 Comparison of Resource Management Organizations**

RM Organization	Control	Site Autonomy	Fault Tolerance	Scalability	Availability
Centralized	Centralized	Low	Low	Low	Low
Hierarchical	Centralized and Hierarchical	Low	Medium	High	Medium
Decentralized	Distributed	High	High	High	High

## 1.4 Types of grid

Grid can be classified into following types based upon the services offered by grid:

- a) Access Grid,
- b) Application Service Grid,
- c) Computational Grid,
- d) Data Grid,
- e) Data Centric Grid,
- f) Interaction Grid,
- g) Knowledge Grid and,
- h) Utility Grid

Based upon operating system, memory space, number of resources, CPU speed, architecture types, and so on, Grid can be classified into heterogeneous and homogeneous grid.

## 1.5 Resource Discovery Techniques

There are mainly two types of techniques for resource discovery. These are as follows:

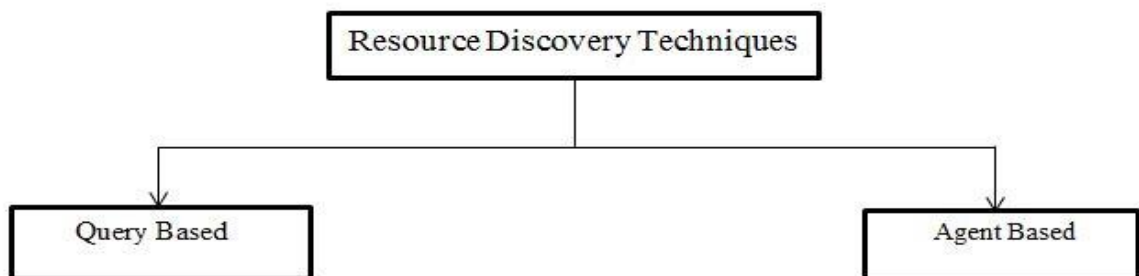
- a) Query based resource discovery,
- b) Agent based resource discovery.

### 1.5.1 Query based resource discovery

A query is generated and sends toward the database to check the availability of resource(s). To take discovery decision, it makes use of fixed query engine.

### 1.5.2 Agent based resource discovery

This technique makes use of agents for query process. Agents are intelligent and autonomous software entities. Agent works on user's behalf and interacts with its surroundings to carry user requests. Agents send code fragment to various nodes for local process, and take discovery decision with the help of internal logic.



**Figure 1.3 Taxonomy of resource discovery techniques**

### REVIEW OF LITERATURE

---

**Muhammad Bilal Qureshi et al., (2014)**, [1], discussed about various Resource Allocation mechanisms. Firstly, they discussed the whole process of Resource Allocation that it consists of resource discovery, resource searching, and job execution, code transfer, data transmission, and monitoring phases. They discussed about resource management strategy to overcome scalability, manageability as well as availability issues. In this strategy, nodes are arranged in peer-to-peer grid organization. Each peer can act as server as well as client at the same time. These nodes (peers) are having logical connection between them in addition to the physical links at underlying network. The resources are discovered by using name lookup strategies. There are two cases to locate resources:

1. If peer knows routing information to locate other peer, then this peer can communicate with other peer directly,
2. Otherwise, information propagation strategy is used. In this case, resources are located by using two main strategies:
  - a) Indexing
  - b) Flooding

**Indexing** is used in unstructured peer-to-peer grid organization. It makes use of Distributed Hash Table (DHT). In this table, hash function is used for indexing purpose.

**Flooding** is used in structured peer-to-peer grid organization. In this strategy, each node propagates information of its local resources. Each node matches user query with its local resources. If user query is matched with local resources, then this information is returned back to the user who initiates this process. Otherwise, query is forwarded to next peer.

The authors also discussed about resource allocation problem that it is represented as quadruple  $(R, A, X, O)$ . Here  $R$  is the set of  $m$  available resources,  $A$  is the set of  $n$  tasks competing for resources,  $X$  is  $m \times n$  matrix in which each entry represents the portion of  $i$ th resources allocated to  $j$ th task, and  $O$  is the objective function.

Then they discussed about various resource allocation mechanisms. Resource Allocation mechanisms play very important role in allocating most appropriate resources to the applications. These mechanisms allocate tasks to the resources to ensure QoS of the application. Different QoS parameters are storage capacity, network bandwidth, and processor utilization. Sometimes, resources are allocated dynamically means resources are allocated to tasks as soon as they discovered. These types of resource allocation mechanisms are known as dynamic resource allocation mechanisms. The authors classify resource allocation mechanisms into three main categories:

1. Centralized Mechanisms,
2. Distributed Mechanisms,
3. Hybrid Mechanisms.

The grid services that are provided by resource allocation mechanisms are:

1. Resource Monitoring,
2. Resource Scheduling.

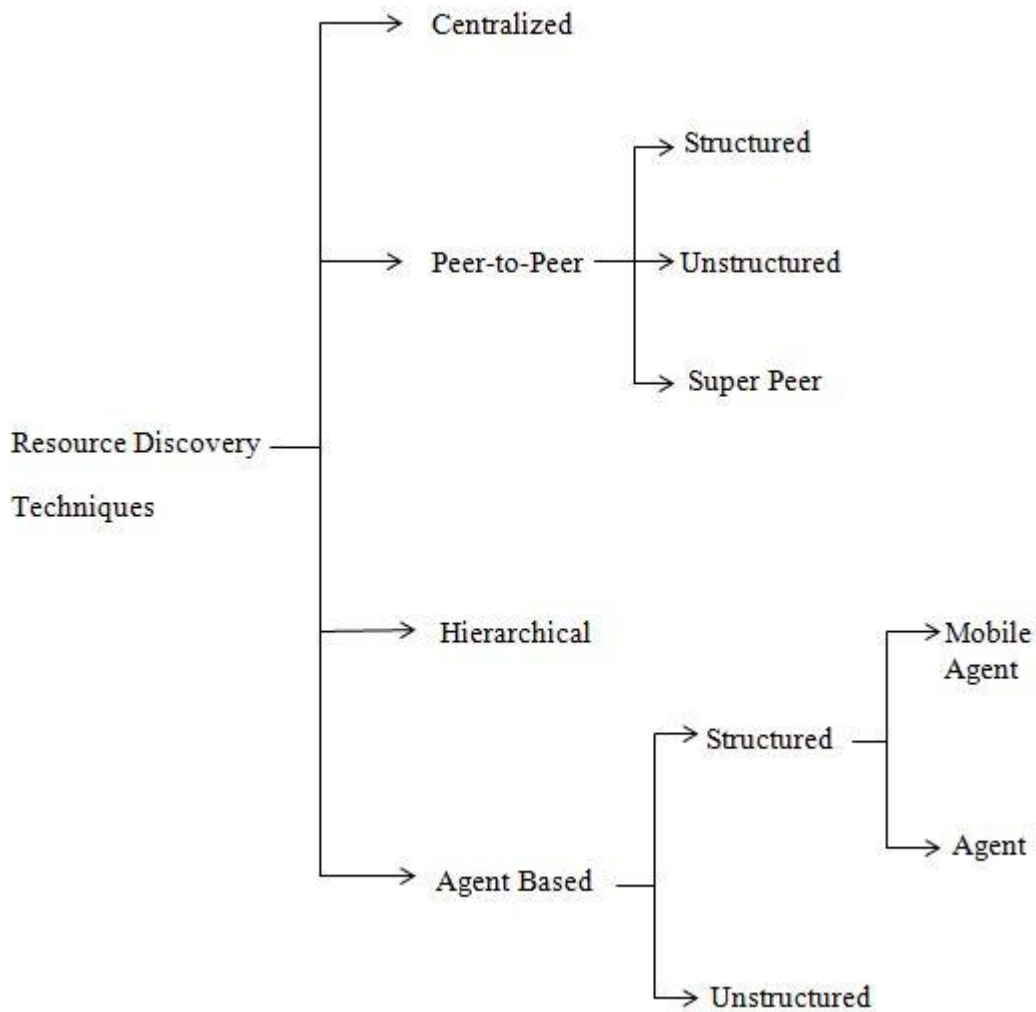
**Mohammed Bakri Bashir et al., (2011), [2]** discussed about various resource discovery techniques for grid computing. They defined resource discovery as a process of locating and seeking suitable resources to execute given tasks in reasonable time regardless of dynamicity and heterogeneity of grid resources. They also discussed that success of other functions of resource allocation highly depends upon the success of resource discovery. There are basically two essential criteria for designing competitive schemes for resource discovery:

1. **Scalability:** - The technique should be scalable with increase in the number of grid resources and users. The performance of static techniques decreases with increase in size of grid.
2. **Reliability:** - This factor is important to consider when failure rate is high. The failure can be server failure, or false positive errors due to TTL (Time to Live) limitation.
3. **Dynamicity:** - The dynamic behavior of central server has great effect on reliability of system, if central server is single point of failure.

They classify the resource discovery techniques in four main categories. These are as follows:

1. **Centralized Technique:** - In this technique, centralized database is used to store the status information of all grid resources. This technique is easy to implement and cost effective. But there is single point of failure and can create bottleneck when large number of user queries are there.
2. **Hierarchical Technique:** - The information services are distributed at various levels. There is control database server at each level which handles update requests from resources. It is more scalable than centralized technique and also reduce bottleneck problem. But having a single point of failure point.
3. **Peer-to-Peer Technique:** - It is the form of decentralized organization. Each peer can act as client as well as server at one time. Large number of nodes can participate.
4. **Agent Based Technique:** - Due to autonomy property of agents, this technique is highly used in grid. To locate other migration site, agents use their own migration policies.

The taxonomy of resource discovery techniques is following Figure 2.1:



**Figure 2.1 Types of Resource Discovery Techniques**

**Saeid Saryazdi et al., (2009)**, [3] discussed that classical optimization algorithms are failed to solve optimization problems which are having high dimensional search space. Therefore, they proposed a new optimization algorithm which is based on law of gravity. They named it as ‘Gravitational Search Algorithm’. It is based on Newton law of gravity which states that ‘Every particle attracts every other particle with a force which is directly proportional to the product of their masses and inversely proportional to the distance between them’. This algorithm comes under the category of population based algorithms. The two common aspects of these algorithms are:

1. **Exploration:** - It is defined as the ability to expand the search space
2. **Exploitation:** - It is defined as the ability to find optimum solution from the list of good solutions.

As the time proceeds, Exploitation fades in and Exploration fades out. During iteration, algorithm passes the following three phases:-

1. Self-Adaptation
2. Cooperation
3. Competition

In proposed algorithm, objects are agents and their masses are used to measure the performance of agents. These objects attract each other with the help of gravitational force and this force causes movement of objects towards the heavier objects. The heavier objects are other objects which are having heavier mass and these masses provide good solutions. The heavier masses move slowly than the lighter masses. This thing corresponds to the exploitation step. In GSA algorithm, each agent is specified by four specifications: position, active gravitational mass, passive mass, and inertia mass. These masses are calculated by using fitness function and position represents the solution. Each mass (agent) provide solution. As time proceeds, algorithm navigates by adjusting gravitational and inertia masses. Therefore, lighter masses will attract toward the heaviest mass. Thus this mass presents the optimum solution.

**Amirreza Zarrabi et al., (2013), [4]** proposed one of the population based metaheuristic algorithm i.e. GSA (Gravitational Algorithm) to schedule task(s) in computational grids. They used GSA algorithm to obtain better solutions than other metaheuristic algorithms like genetic algorithm (GA) and particle swarm optimization (PSO) algorithm by minimizing makespan and flowtime of grid. In this paper, ETC matrix is used to represent execution time of different task(s) on different machine(s). It is  $m \times n$  matrix where  $m$  is the number of tasks and  $n$  is the number of machines. Each entry  $(i, j)$  represents the expected execution time of task  $i$  on machine  $j$ . Two main criteria that are used to evaluate the performance of algorithm are:

1. **Makespan:** - It is defines as the time when grid finishes the latest task
2. **Flowtime:** - It is defined as the average response time of the task.

Minimizing makespan means average task should be finish as soon as possible so that long tasks can take more time. Whereas, minimizing flowtime means no task will take too long time to finish. In this paper, authors use one of the criteria based upon the value of  $\lambda$ . Fitness function is used to update masses of agents.

$$\text{Fitness} = \lambda \times \text{makespan} + (1-\lambda) \times \text{flowtime}/ m$$

Better solutions will gain mass and worst solutions will loss mass. Heavier masses pull other masses because they are having higher attraction force. Each agent represents the solution in the form of  $m \times n$  matrix where  $m$  is number of machines and  $n$  is number of tasks. Each entry  $(i, j)$  represents whether the task  $j$  is allocated to machine  $i$ . They used min-min heuristic to generate one agent and others are generated randomly. Then they calculate fitness function by considering makespan as their objective. The agent which is having least value for fitness function will be selected as final candidate.

**Belabbas Yagoubi et al., (2011), [5]** examined static and dynamic task assignment methodologies for dependent tasks. Their objectives are: minimizing average response time of task(s), reducing communication costs by using static and dynamic methods for task placement. They discussed that task assignment problem involve the issue of utilizing idle nodes that are scattered across different geographical regions. The main goals of task assignment problem are:

- a) Enhancing performance,
- b) Resource sharing
- c) Extensibility,
- d) Increase availability.

Then they discussed various assignment algorithms. The assignment algorithms are categorized in two main categories: single factor assignment algorithms and multiple factor assignment algorithms. They discussed that dependent tasks can be represented by DAG (Directed Acyclic Graph). In their proposed strategy, they used hybrid approach to schedule tasks. In static task assignment, task is assigned to appropriate computing element, whereas in dynamic case, system will be adjusted dynamically by considering clusters workload. Firstly, they divide the DAG in  $n$  connected components in static assignment phase and assign these to various cluster managers which further schedule



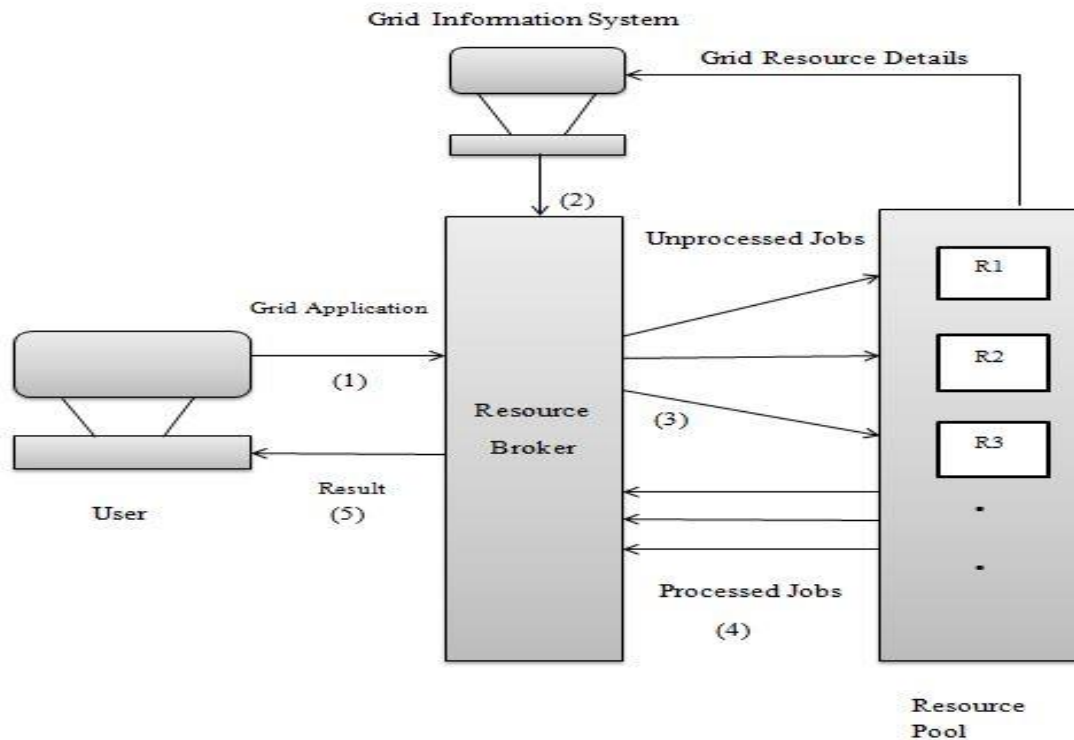
them to computing element by using round robin or other strategy. Then in dynamic task placement strategy, each computing element runs the first entry task and updates the connected component. Also computing element executed the connected component algorithm to determine new entry task and computes its execution time. Then this information is propagated to cluster manager and other computing nodes. If CE examined that it is more loaded, some of the connected components shifted to other computing elements. Also the information of this transfer will send to cluster manager.

**Manvi S.S et al., (2005),** [6] discussed that there are three types of agents which will work in resource allocation mechanisms:

1. Resource Brokering Agents,
2. Job Agents,
3. Resource Monitoring Agents.

Resource Brokering Agents are used to schedule resources. Resource Brokering Agents act as resource scheduler. They also act as broker for submitting unscheduled jobs to resources. RBAs allow users to submit their jobs with the help of Job Agents. The basic grid model is shown in Figure 2.2

Job Agents are used to search resources by sending code fragment to all the resources. Then, the decision is made on the basis of internal logic. Resource Monitoring Agents reside inside each node of the cluster. These agents will inform the cluster manager about the status of resources.



**Figure 2.2 Basic Grid model**

**K. Somasundaram et al., (2009), [7]** proposed a new dynamic scheduling algorithm known as Swift Scheduler. This algorithm is the combination of heuristic algorithm and traditional Shortest Job First algorithm. They will consider memory requirement of task, CPU requirement as well as priority of the task. Their main objective is to reduce the waiting time of the task(s) in job queue as well as to reduce overall computational time. The proposed algorithm works in the following steps: Different users give tasks and these incoming tasks are collected and stored in job list. The available resources are collected and stored in resource list. By running the swift scheduling algorithm, tasks in job list are mapped to resources in resource list. These resources are selected by using some heuristic function. The function selects the optimized resource for executing particular task which completes the task in minimum time. The authors use the swift scheduler of GridSim and compare its performance against the other algorithms like First Come First Serve (FCFS), Shortest Job First (SJF) etc. They proved that Swift Scheduler completed all the tasks with minimum completion time and minimize cost by utilizing all the resources in efficient manner as compared to the other schedulers.

**D.Maruthanayagam et al., (2011)** [8] proposed an Improved Ant Colony Scheduling technique by combining Ant Colony Optimization with the concept of Resource Aware Scheduling Algorithm (RASA). According to the proposed technique, the first thing need to do is to select a set of computers and network connection for an application. To estimate the completion time of the tasks on each of the available grid resources, they used a task algorithm of Resource Aware Scheduling Algorithm. Then they decided to apply the Max-Min and Min-Min algorithms. The expected execution time of each task on each machine is represented by Expected Time calculation. The Min-Min algorithm starts with the unmapped set of tasks. It computes the minimum completion time for each unmapped task. Then it selects the task with overall minimum completion time and assigns it to the corresponding resource. This process repeats until all the tasks are mapped. The Max-Min algorithm starts with the unmapped set of tasks. It computes the minimum completion time for each unmapped task. Then it selects the task with overall maximum completion time from minimum completion time and assigns it to the corresponding resource. This process repeats until all the tasks are mapped. The Ant Colony Optimization technique is used to find the shortest path between the nest and the food. In RASA, if number of available resources is odd, it allocates the resource to first task by Min-Min or Max-Min. The remaining tasks are assigned by using one of these techniques alternatively. This alternative interchange of these two techniques results in consecutive execution of small and large task(s). The Ant Colony Optimization (ACO) and Resource Aware Scheduling Algorithm (RASA) are combined to optimize workflow execution time. The scheduling algorithm is executed periodically. To form the ET matrix and to start scheduling, the algorithm finds all the available resources at run time. When all jobs are dispatched, the scheduler starts scheduling the unscheduled task(s). This guarantees that all machines are fully loaded at maximum times. They compared the proposed technique with existing ACO and results showed that proposed technique is better than existing ACO in terms of minimum makespan time.

**T. Stutzle et. al., (1997), [10]** proposed MMAS i.e. Max Min Ant System technique. It is also a heuristic based technique. It is a hybrid technique in which MMAS and local search method were combined together. Local search method is used in quadratic assignment problem. Max Min Ant System used greedier search approach than Ant System. It strongly exploits the search space. It adds the pheromone to the best solutions when

updating the pheromone trail. Weak solutions are discarded to be updated. By adding the local search algorithms, MMAS can be easily extended. To exploit best solutions during iteration, it considers that only one ant will add the pheromone during pheromone update. This ant may be the one which found best solution from beginning or in the current iteration. To achieve higher exploitation, it initializes the pheromone to max interval. By depending upon the instance type, it strongly impacts the performance. It was noticed that MMAS gives good results than Ant System for quadratic assignment, travelling salesman problem, and resource allocation in grid.

**Meriem Meddeber et al., (2011)** [11] proposed a Static Task Assignment technique for dependent jobs. Their goal is to first reduce average response time of tasks whenever possible. Then to reduce transfer cost by taking into consideration the dependency constraints. In static task assignment technique, simple information of system is used to distribute tasks. These tasks are distributed by making use of mathematical formulas or other methods. The tasks are distributed in such a way that every node or resource can process the task(s) until completed. According to the proposed technique, they first need to form clusters of initially collected nodes by using distributed clustering algorithm (DCA). DCA uses two types of messages: - Ch ( $v$ ) and Join ( $v; u$ ). Every node starts the execution of algorithm at same time by executing Init procedure. The working of Distributed Clustering Algorithm (DCA) is shown in Figure 2.3.

The next step is scheduling of tasks. Each clusterhead uses Heterogeneous Earliest Finish Time (HEFT) algorithm. According to this algorithm, each clusterhead uses static assignment of dependent tasks on a heterogeneous platform. In this way all dependent tasks of whole application is assigned to heterogeneous resources. The advantage of this technique is its simplicity and to make good schedules by minimizing makespan. HEFT first creates list of tasks based on their priority and then it makes local optimum decisions for each tasks on the basis of estimated finish time. Heft algorithm works in three phases:

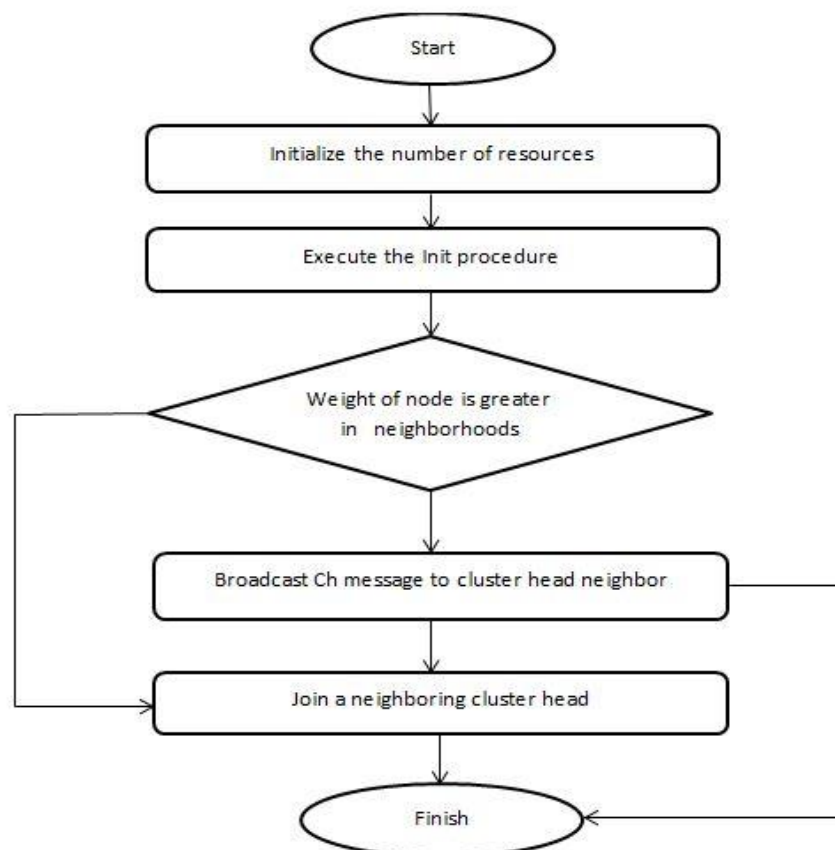
-

- i. **Weighting:** - In this phase, the weights are assigned to nodes based on the expected execution times of the tasks. The weights to the edges are assigned based on the expected data transfer times between the resources. HEFT assumes that these times are known. Various methods can be used to predict these times

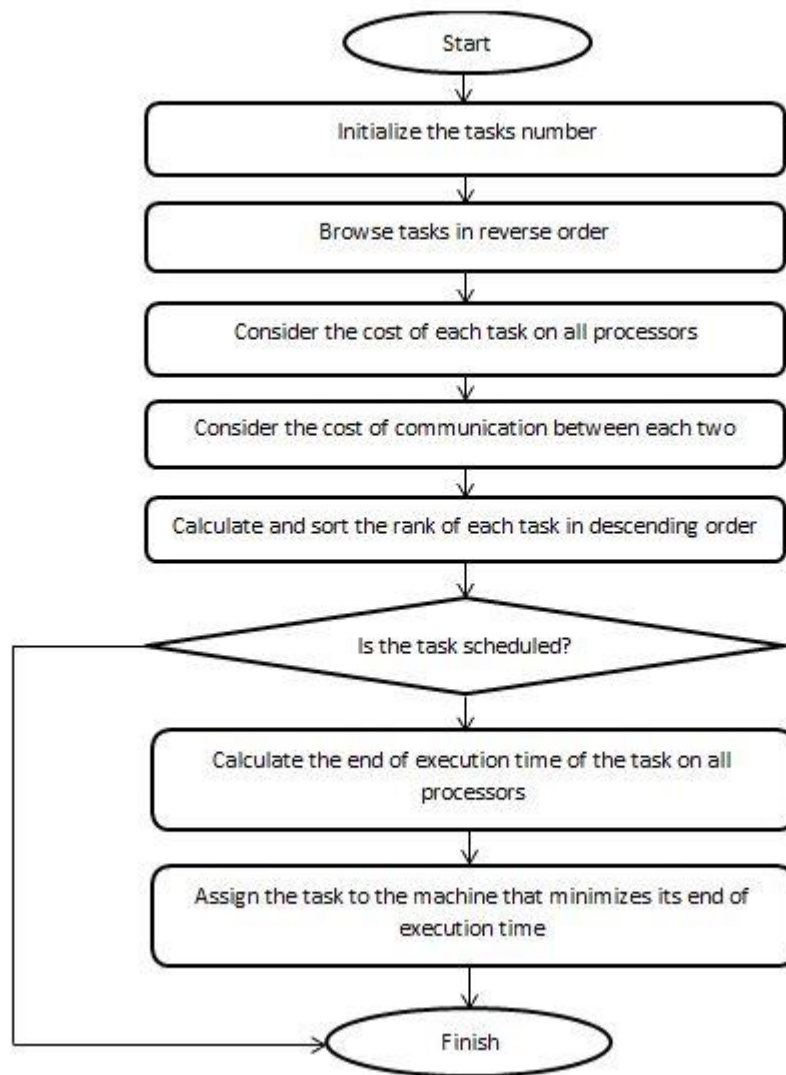
but the most common method is to take average of the times on all resources for each task.

- ii. **Ranking:** - In this phase, rank value is assigned to each task by traversing the graph in backward direction. The higher rank value means the higher priority. The rank value of a task is equal to the task's weight by adding it with maximum successive weight. Then the tasks are sorted in decreasing order based on the rank values.
- iii. **Mapping:** - In this phase, tasks is mapped to the resource which minimize the task's earliest expected finish time. All the tasks are mapped in this way to their corresponding fitted resource.

The working of HEFT algorithm is shown in Figure 2.4. Then they compare the results of proposed technique with random strategy, HEFT strategy (without using DCA) and strategy based on Meta tasks.



**Figure 2.3 Distributed Clustering Algorithm**

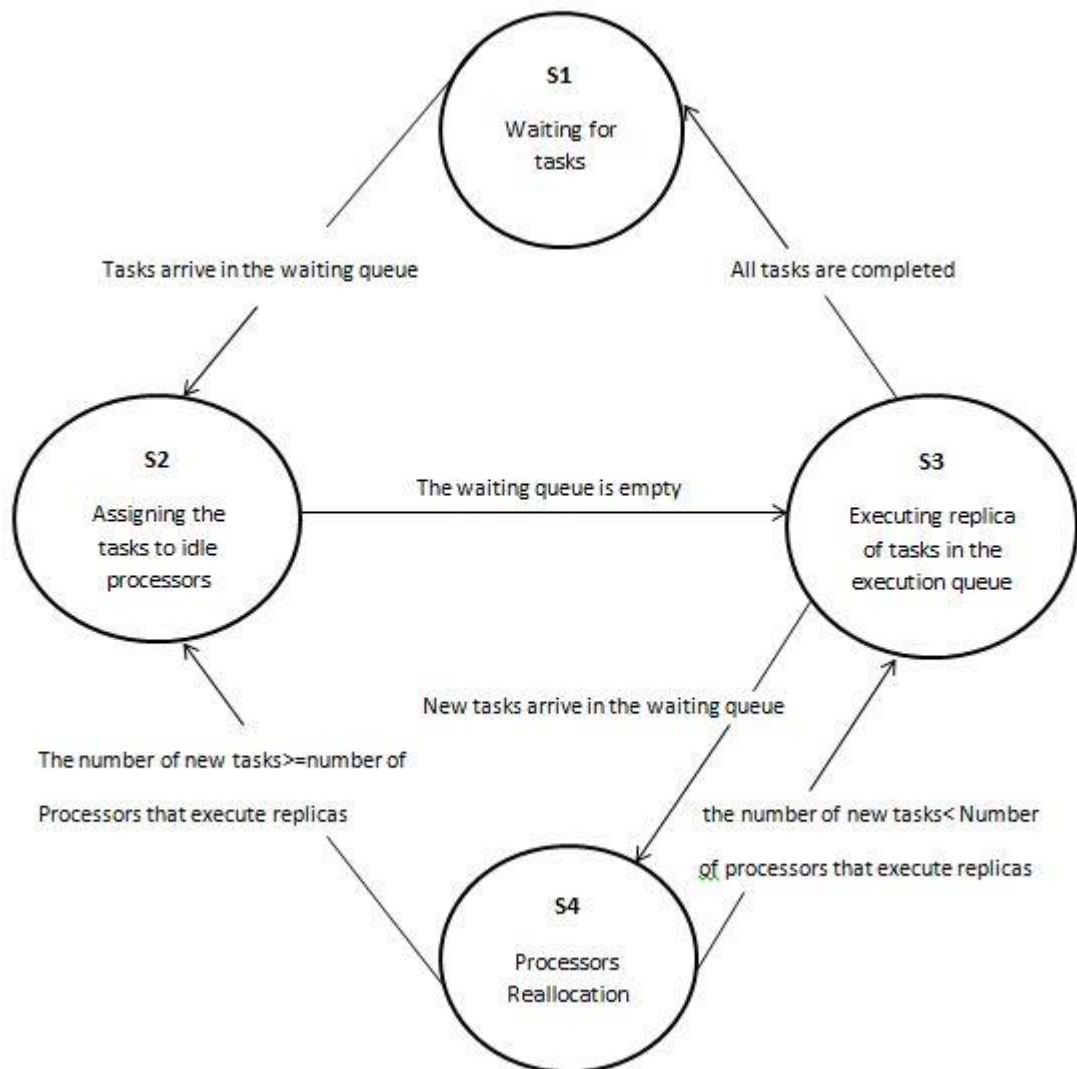


**Figure 2.4 Heterogeneous Earliest Finish Time**

**H. Yan et al., (2005).** [12] proposed Improved Ant Algorithm which takes the idea from basic Ant Colony Optimization (ACO) for scheduling jobs. To update pheromone values, it uses encouragement, punishment coefficient and load balancing factor. These coefficients are defined by the user itself. When pheromone value is calculated, we need to consider the status of each resource. The job(s) is allocated to the resource having the highest value of the pheromone. If the assigned job is completed successfully by the resource, then the encouragement coefficient to added to the pheromone value. In this way, pheromone value of the resource is increased to assign the next job. If the assigned job is not completed successfully by the resource, then it will be punished. It decreases

pheromone value of that resource by adding punishment coefficient. The balancing factor is used to change the pheromone values by considering the load on each resource.

**Sunita Bansal et al., (2011)** [13] proposed a novel scheduling technique to schedule tasks dynamically and adaptively without having need of prior information of incoming tasks. The proposed approach considers the grid environment as state transition diagram. Then a prioritized round robin algorithm with task replication is used to schedule the tasks. It makes use of prediction information on processor utilization for each individual node. They represent their approach by considering that grid can occupy one of four states (shown in figure 2.5) at any given interval of time. They used two types of queues in their simulation: -



**Figure 2.5 Four different states of grid systems**

1. **The waiting queue** consists of the tasks which are waiting to be mapped to their corresponding machines. This queue is implemented by First in First out (FIFO). The task which arrives first is the head of queue and is mapped before all other tasks in the queue.
2. **The execution queue** consists of the tasks which are currently executing. It is implemented as a circular queue. In this queue each task has some specific order which is not similar to the order of other tasks. It uses three pointers to scan the circular list:-
  - a) **Current Pointer:** - It is used to point a task with the highest priority at current instance of time,
  - b) **Next Pointer:** - It is used to point the task having second highest priority. It is placed next to the current task in clockwise direction.
  - c) **Last Pointer:** - It is used to point to task having least priority and this task is placed besides to the current task in the anti- clockwise direction.

In **State 1**, idle scheduler waits for tasks. Both queues are empty initially. When number of incoming tasks crosses the threshold value, transition shifts to state 2. In **State 2**, execution queue is initially empty but waiting queue contains the incoming tasks. The tasks are mapped to resources in idle list one by one starting from the header task in the waiting queue. As we allocate the resource from idle list, that resource will be removed from idle list. The mapped task moves from waiting queue to the execution queue. They implement the logic to give highest priority to a task which was mapped to the slowest processor and vice versa. This is the task which will be replicated. So by replicating this task there would be the high probability that the new machine will complete this task before the already assigned resource. In **State 3**, the execution queue comprises of tasks currently executed and the waiting queue is initially empty. If task will be completed by the machine, the processors and all the machines those were assigned to the completed task would be released. This task will be removed from the execution queue. Here is the need of updating processing power of the released machine. The processing power depends upon the number of instructions executed by this machine for previously completed task and the time it took to complete the task. If it is greater than maxProcSpeed of the task which is pointed by the current pointer, then this machine



needs to execute the replica of current task. If number of tasks in waiting queue exceeds threshold before completing the execution of all the tasks in execution queue, then transition shifts to state 4. In State 4, both queues are non-empty. There are two scenarios at this stage: -

1. The number of machines which execute replicas is less than the number of the tasks in the Waiting Queue.
2. The number of machines which execute replicas is more than the number of the tasks in the Waiting Queue.

The tasks are traversed in an anticlockwise manner, starting from task which is pointed by last pointer. If this task has more than one machine to be allocated then the processor which is at the tail end will be freed and assigned the task at the head of waiting queue. In Case 1, we stop the traversal if all the tasks in execution queue one and only one machine and transition shifts to State 2. In Case2, we stop the traversal if machine(s) is assigned to all the tasks in waiting queue and transition shifts to State 3. They compare the results of proposed prioritized round robin algorithm with round robin technique.

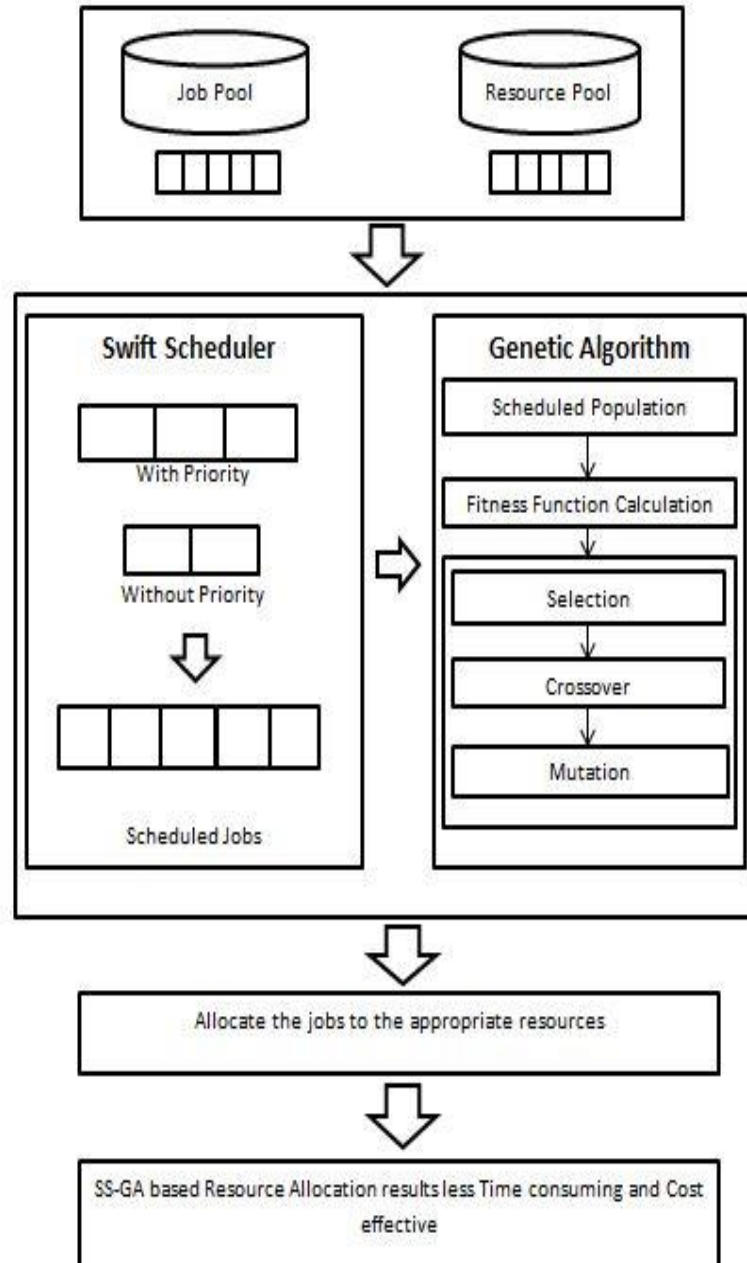
**C. Blum et al., (2005)** [15] proposed a heuristic based approach known as Ant Colony Optimization (ACO). It is population based technique. It is inspired from the behavior of ants while they search for their food. When ants need to search for their food, they start moving randomly by laying pheromone trail in their path. Then the remaining ants in ant colony works together by following pheromone trail(s) laid by the fellow ant(s) to find the shortest path from their nest to food. The pheromone trail is chemical substance which is released by ants when they move. The path having high pheromone value is considered to be the shortest path. Over the time, the pheromones will start evaporating. This evaporation process reduces attractive strength of the path. Thus the density of pheromones is higher on the shorter path than the longer path. This algorithm is effective algorithm to use in resource allocation in grid computing. It starts by computing the initial pheromone value for all the resources. Then pheromone value will be updated for each resource. Then probability will be computed to choose the resource. The resource with the highest probability will be considered as fittest resource to be allocated to new incoming job.

**K. Satish et al., (2013)** [16] proposed a technique named as SS-GA by collaborating Swift Scheduler and Genetic Algorithm based resource allocation. The main objective to propose this technique is to allocate fittest resources to proper jobs to meet the QoS (Quality of Service) requirements i.e. to minimize job completion time, resource utilization, cost minimization and economy. They proposed this new technique to make the process of resource allocation more proficient than other methods. In their methodology, they take resources and jobs to be processed as input data sets. A resource pool consists of number of resources with their ids, capacity, and cost to execute particular job(s). A job pool consists of number of jobs with their ids, length and priority of job. The Swift Scheduler (SS) collects the job(s) from different users and then swift them either by their priority or by their length. Then it allocates the resources to jobs based on their swift according to their priority in the job pool. The Genetic Algorithm (GA) is a population-based technique. It allocates the resources to the jobs by evaluating the fitness function. In their methodology, they considered that there are  $n$  number of jobs to be processed, and there are  $m$  number of resources which needs to process these jobs ( $n > m$ ). They also consider some assumptions to make their resource allocation technique as the finest technique. These assumptions are as follows: -

- i. Every job is independent to each other,
- ii. Every job is assigned priority among them,
- iii. The jobs can't migrate,
- iv. At early stage, every resource and job can be simultaneously available,
- v. Every job and resource has its unique id to uniquely identify and to avoid conflicts while processing,
- vi. In job and resource input data sets, each attribute such as job priority, job length, resource capacity and cost should need to be specified to make resource allocation effective,
- vii. Each resource can process only one job at a time and no interruption is permitted before the completion of job.

In the proposed SS-GA algorithm, they combine priority based job scheduling which is offered by Swift Scheduler (SS), and powerful accurateness finding ability which is

offered by the Genetic Algorithm (GA). In this way, they made full use of advantages of both techniques. Figure 2.6 shows the proposed SS-GA algorithm for Resource Allocation.



**Figure 2.6 Proposed SS-GA Algorithm for Resource Allocation**

**K. Mahamud et. al., (2010)** [17] proposed hybrid approach for scheduling jobs to grid resources. It combines the Ant System and Min-Max Ant System. It uses the local

pheromone trial update. To set the restriction on pheromone evaporation it limits trial values. To calculate the pheromones it focuses on the status of grid resources. Therefore to store the status of available resources, it makes use of a matrix. Agents are used to update the resource table.

**R. Saxena et.al., (2015)** [18] proposed a heuristic technique named as AHSWDG(An Ant Based Heuristic Approach to Scheduling and Workload Distribution in Computational Grids) which is based on Ant Colony optimization for balanced distribution of workloads. Firstly it computes initial computational capacity of all available resources, and then it computes the probability for all the resources to allocate incoming job(s). The resource with highest probability is the one to allocate to incoming job. After this most optimal resource will find by the algorithm which is the one which successfully completed the job. If found job will be assigned to it and resume the job on this resource. But here the one problem is that if the job failed on no optimal resource would be found then this failed job again put into the actual job queue. No technique is used to recover the failed job(s). It is better than the random approach in utilizing the resources.

#### 3.1 Scope of the study

Dynamism of the jobs and the resources is a major concern in Grid, so a good scheduling algorithm will lead to efficiently allocating the resources to the jobs. The proposed algorithm schedules the job(s) to the resource(s) in efficient and reliable manner by considering the following factors:

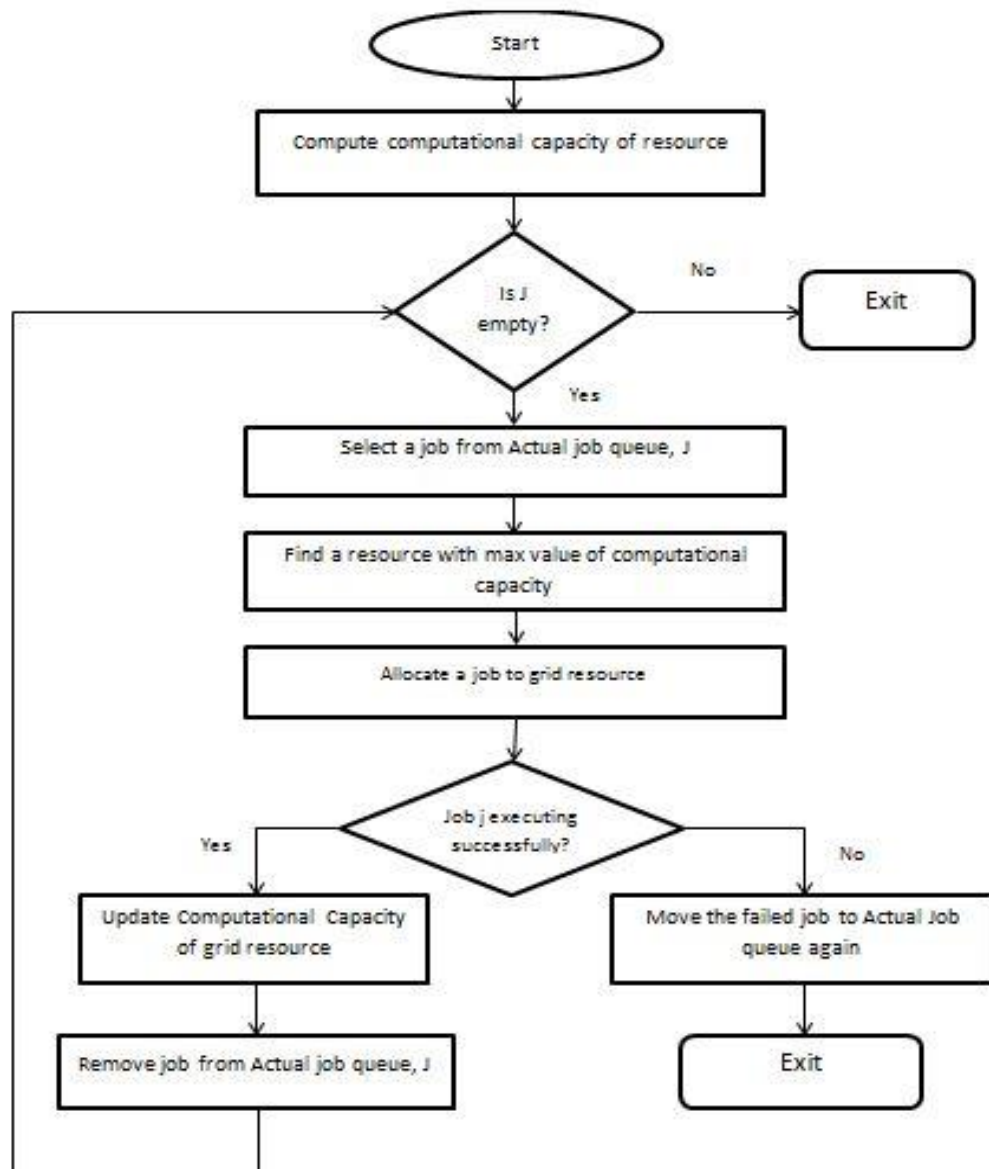
1. Dynamicity of the resources to handle them efficiently as they enter and leave the grid environment,
2. Because of large number of resources, finding the best resource becomes difficult and time consuming. According to the type of user (primarily classified as Type A, B, and C), the resources are handled,
3. Resource properties such as CPU, RAM, Hard drive, Bandwidth are primarily taken for scheduling the jobs on the resources,
4. Based upon the above mentioned properties, we will classify the resources into three clusters high, medium and low and then map the users to the respective resource clusters.
5. For fault tolerance a copy of the job will also be scheduled at resources of low cluster.

The proposed algorithm has a great scope in dynamic environment and for doing user based scheduling.

#### 3.2 Problem Formulation

1. My problem definition is “An efficient resource allocation in grid computing environment”.
2. The problem of existing techniques is that to find the fittest resource to execute the new incoming job, we need to search from all the available resources in grid. To overcome this problem, I decide to categorize the resources according to their configuration.
3. Another problem of the existing techniques is that all the users are considered to be of same level. All the users can use same level of services which may again lead to over utilization of resources where it is not needed and may give low performance where it is highly expected. So to overcome this problem, I decide to categorize the users according to their SLA (Service Level Agreement) type.

4. In this way, services will be given to the users according to their type and optimized results can be achieved.
5. Another flaw in existing technique like AHSWDG (shown in figure 3.1) is that it directly sends the failed job to job queue again. Algorithm again needs to load the job, perform computations and to do resource allocation. The proposed technique will cut down this overhead.
6. Firstly the proposed technique will find the reason of failure and then provide fault tolerance by activating the services according to the user priority. In this way, optimized results and reliability can be achieved. Also failure rate can be decreased when comparing with AHSWDG.



**Figure 3.1 Flowchart of AHSDWG**

### **3.3 Objectives:**

The main objective is to develop scheduling algorithm which can optimize resource allocation process in grid environment, and allocating the job(s) to the machine(s) in efficient manner.

1. To improve scheduling of job(s) in grid, algorithm will check which particular machine is to be considered for scheduling the job(s) by considering resource configurations, the user type, and dynamicity of the resources as they can leave or join grid at any time,
2. The resources are handled dynamically according to the user type (primarily classified as platinum, gold, and silver),
3. Resource properties such as CPU, RAM, Hard drive, and Bandwidth are primarily taken as the factors for giving priority to the resources, and accordingly they will be classified into three categories: High, Medium, and Low end cluster.
4. Then we map the users to respective resource cluster.
5. To increase the efficiency, search time of resources will be decreased by clustering resources efficiently, and by dividing the users into different categories.
6. To provide fault tolerance, a copy of the job will be scheduled at the resource(s) of the low cluster

### **3.4 Research Methodology**

The main motive of proposed technique is to schedule job(s) in grid environment efficiently and to provide fault tolerance to handle different types of failures. How?

1. To increase the efficiency of the proposed technique, I decide to make clusters of resources in efficient manner so that the search time of resources can be decreased.
2. Three types of logical clusters should be made and are as follows:-
  - i. High end resources,
  - ii. Medium end resources,
  - iii. Low end resources.
3. These clusters will be formed on the basis of the following characteristics of resources:
  - i. CPU (MIPS rating),
  - ii. Memory,

- iii. Bandwidth,
  - iv. HDD.
4. Let's suppose we have 1000 resources out of them some are clustered as high, medium or low configuration resources. If the job is submitted by the high category user, then instead of searching for the best resource out of 1000 resources, the algorithm will search for the resources in highly configured cluster which in turn will help to reduce the search time.
  5. For doing user based scheduling, users will be categorized into three categories:
    - i. Type A users,
    - ii. Type B users,
    - iii. Type C users.
  6. The factor which needs to be considered to categorize the users is **charges**; they paid to use grid resources.
  7. How it works?
    - i. When the request for the resources will come from platinum user, we don't need to search fittest node from all resources. We just need to search from high end resources which lead to minimize the search time, hence increase the efficiency.
    - ii. In case, if no high end resource is free or available then only we need to search from medium or low end resources.
  8. To handle the failure, firstly consider three types of failures:-
    - i. **VM failure:** - Job(s) will fail due to some failure in VM. On one host multiple Virtual Machines (VMs) can run.
    - ii. **Host failure:** - Job(s) will fail because host will fail.
    - iii. **Network failure:** - Job(s) will fail because there will be some problem in network connection.

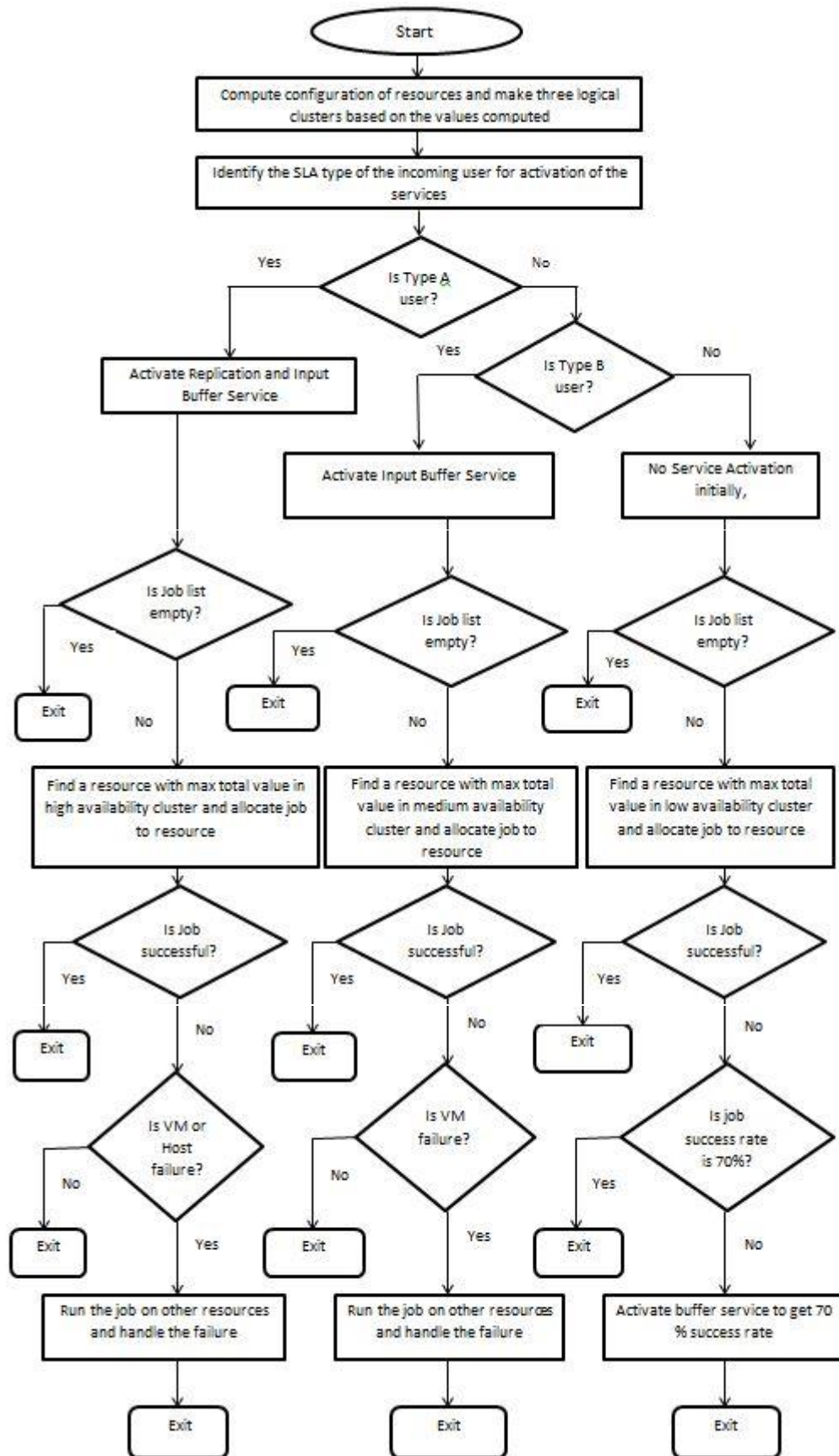
To provide fault tolerance, algorithm will use two types of services:-



- i. **Replication Service:** - This is the service which is used to replicate job(s) of logging user to two or more resources (VMs). Replication will be done on the resource (VM) of other host not on the same host. Each host is divided into 2 or more VMs. Therefore when job will come, copy of this job should also be run on VM of other host. This service is used to handle **VM as well as Host failure**. In this service, double resources would be consumed which is wastage of resources.
  - ii. **Input Buffered Service:** - In this service, buffer (a part of memory) is used to store the incoming jobs at each host before allocating them resources (VMs of host). Input buffer is working as a cache memory and stores the job (i.e. job id and job itself) before processing it at one of resource. Buffer should be implemented at each host to store only jobs that comes to that host. It tackles the job failure due to **VM failure**.
9. Activation of these services totally depends upon the category of user (i.e. Type A, B or C), because not all the users pays equal amount to use grid resource then why to provide all the services to all the users? And why to wastage extra resources for all category of users? How to activate?
- i. If the user is **Type A user**, then **both services will be activated** to handle VM as well as Host failure. As I discuss before that to replicate the job(s), double resources will be used. Therefore replication service will only be activated for Type A users because they pay very high amount to use the resources. Jobs of this category of users will be considered as **crucial jobs**.
  - ii. If the user is **Type B user**, then **input buffer service will be activated** to handle VM failure only.
  - iii. If the user is **Type C user**, the minimum share service would be assured. I am considering that user will be given 70% success ratio when he/she submits the job(s). If this ratio will degrade from 70%, then only input buffer service will be activated for this category of user.
10. As we know grid is dynamic in nature, so machine can leave and join the cluster at any time. To reduce the overhead of grid manager, when new resource wants to

join, it will automatically adjust into one of the cluster (high end, medium end, low end) according to its characteristics.

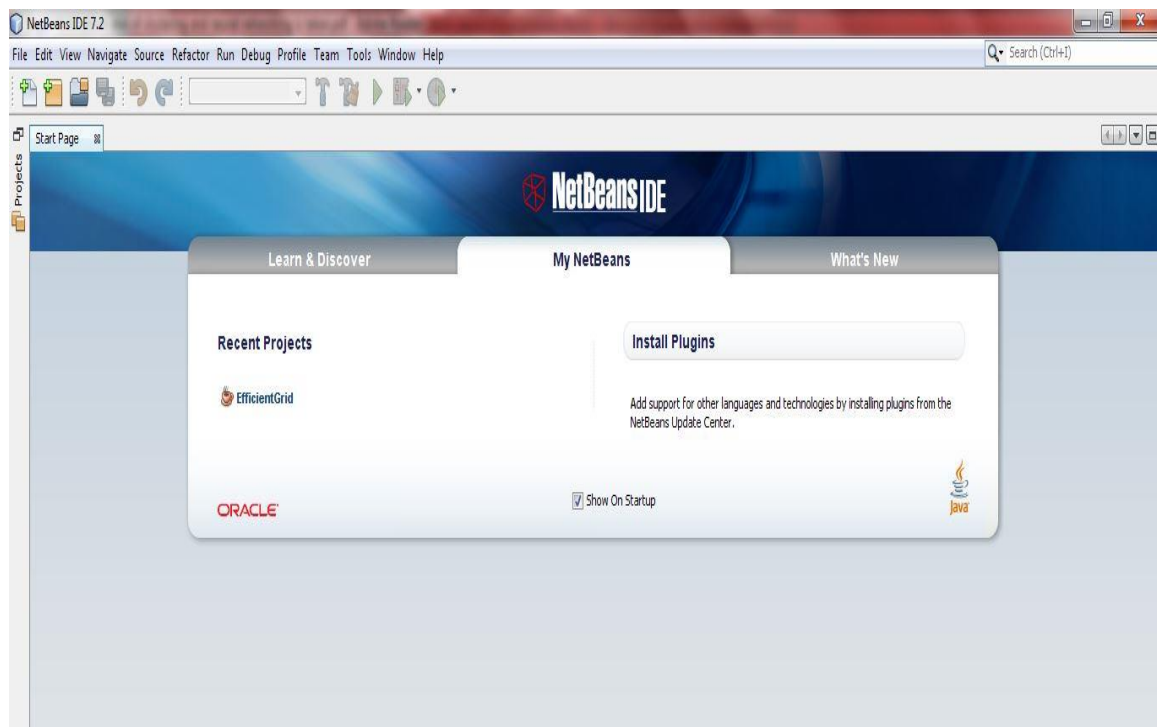
11. The flowchart of proposed technique is shown in figure 3.2



**Figure 3.2 Flowchart of proposed algorithm**

## 4.1 Implementation

The proposed methodology is simulated with the help of Java based Simulation and 'Netbeans IDE 7.2'. Netbeans is a platform where applications are developed using segments called software modules. Fig 4.1 depicts NetBeans development environment.



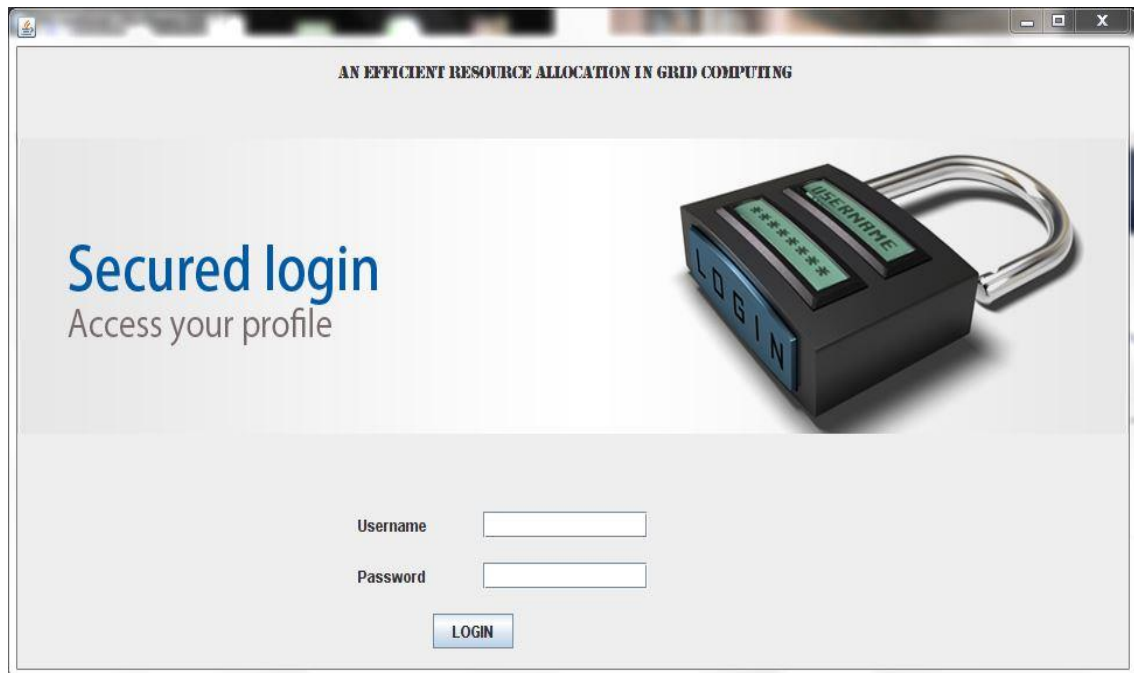
**Figure 4.1 Netbeans IDE**

## 4.2 Experimental Results 1

The first phase in my simulation is Client Login phase. Here the user will login by entering his/her username and password. Then SLA type of user will be recognized. And accordingly activation of services and resource allocation will be done. The whole steps are explained below: -

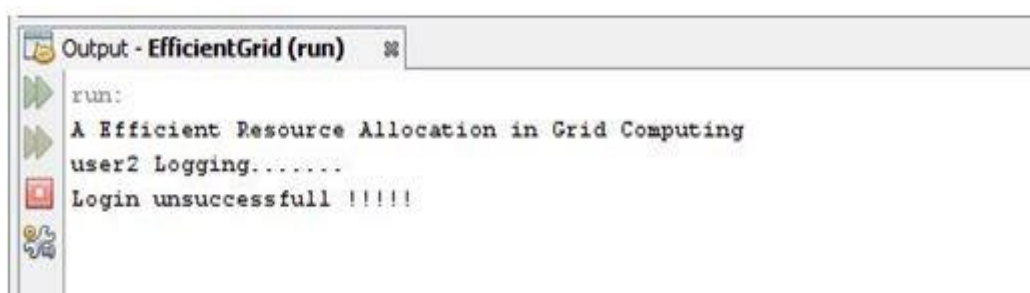
### 4.2.1 Client Interface: -

When we run the project, client interface window will be displayed (shown in figure 4.2).



**Figure 4.2 Client Interface Window**

- i. Here user needs to enter the username and password.
- ii. When user will click on LOGIN button, username and password will be verified from the database at Grid Data Centre to determine the SLA type of user. In this project, these are stored in client.csv file.
- iii. If the username or password is incorrect then following output (shown in figure 4.3) will be displayed and project will stop to proceed.



**Figure 4.3 Unsuccessful Login Output**

- iv. If matched, the project will proceed by determining the SLA type of current user.

## 4.2.2 Grid Master Module

As I already discuss that there are three types of users according to the SLA i.e. Type A, Type B, and Type C. Grid Master activates the services according to the **SLA type**. To provide the fault tolerance and utilize the resources in efficient manner, algorithm will activate the services according to the category of user.

- i. **If user is Type A user**, The window shown in figure 4.4 will be displayed. As we can see that both replication service and Input buffer service will be activated for this category of user(shown in figure 4.5). This category of users paid high amount to use grid resource. The jobs of this category are considered to be crucial ones. Therefore, both VM and Host failure will be handled.

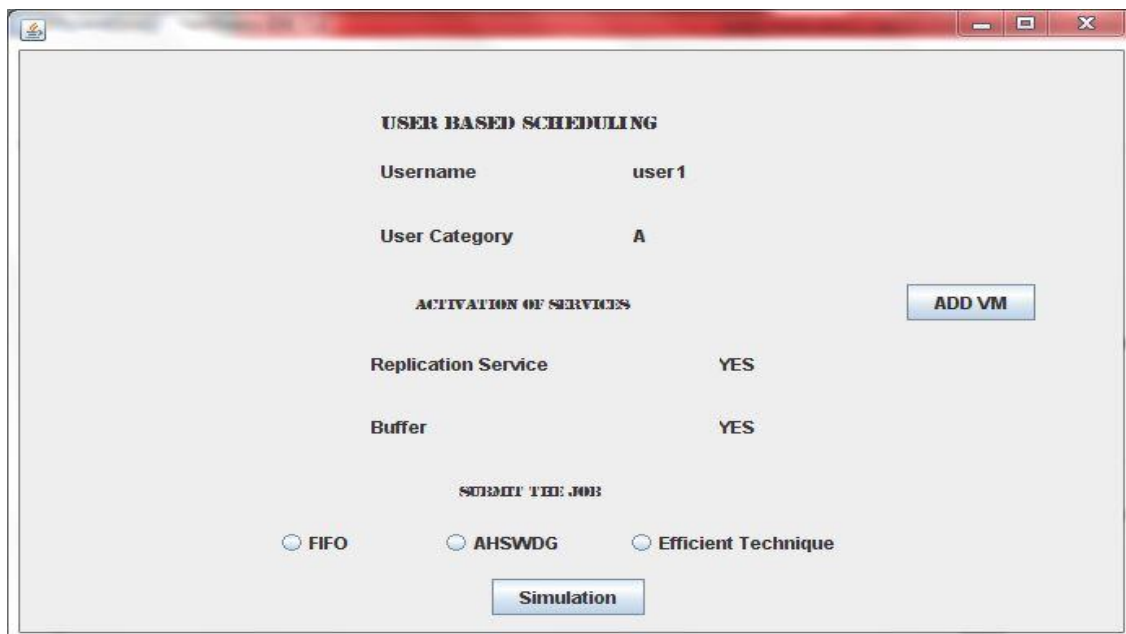


Figure 4.4 Interface for Type A user

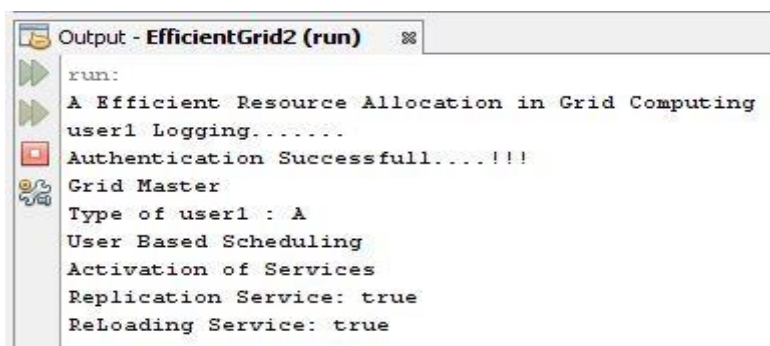
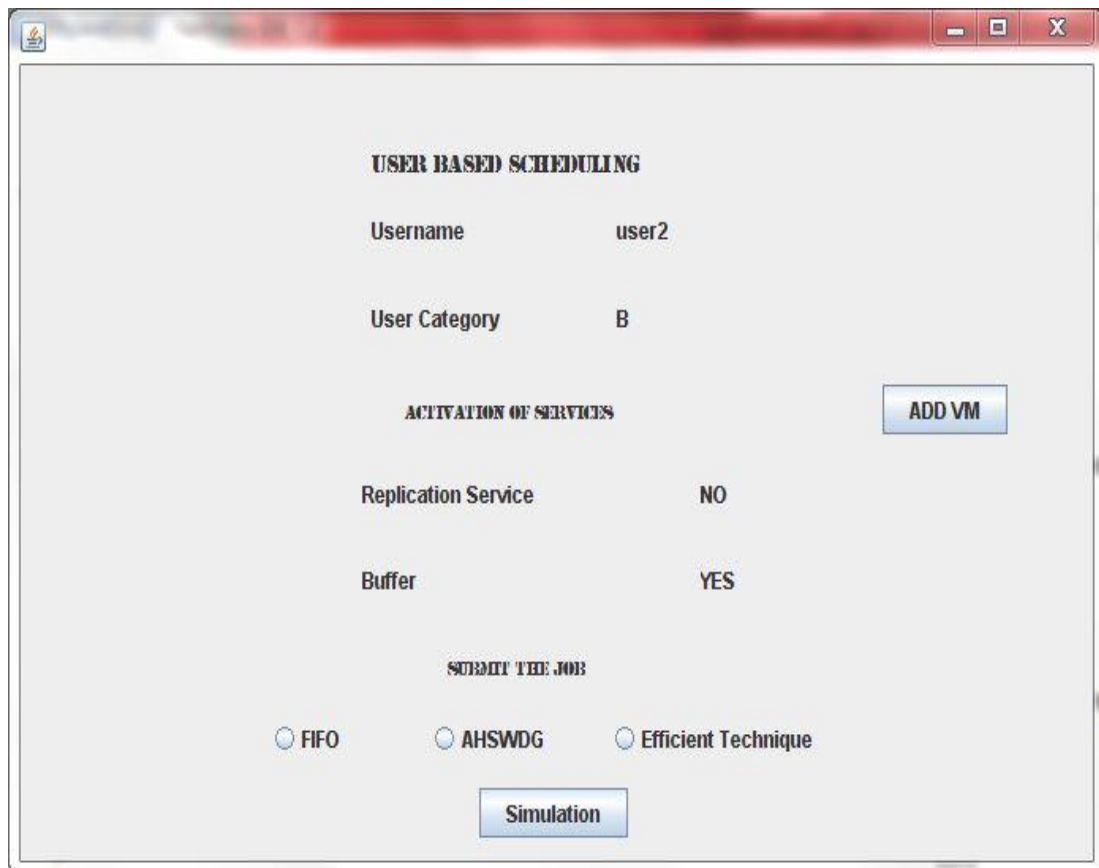


Figure 4.5 Output for Type A after successfully login

- ii. **If user is Type B user**, the window shown in figure 4.6 will be displayed. As we can see in figure 4.7 that only Reloading Service (Input Buffer Service) will be activated for this category of user. Replication Service will not be activated because double resources would be consumed which is the wastage of resources. These are the users which paid medium amount, therefore no need to use double resources for this category. For this category of user, VM failure will be handled.

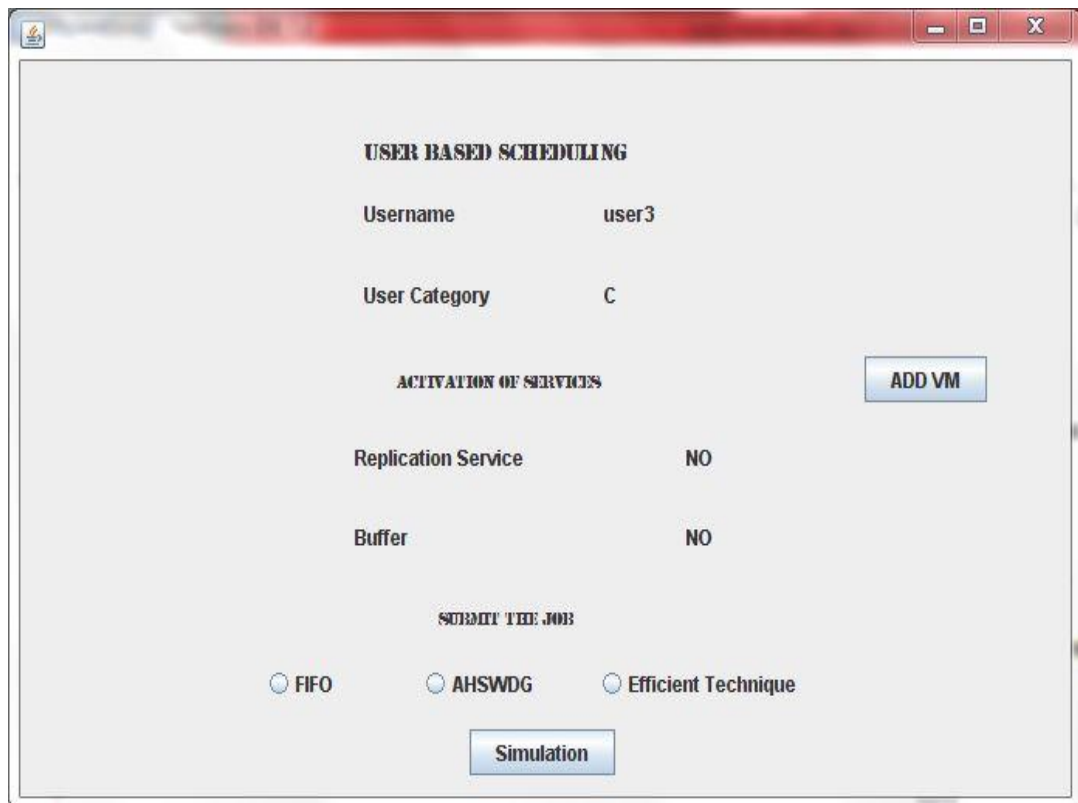


**Figure 4.6 Interface for Type B user**

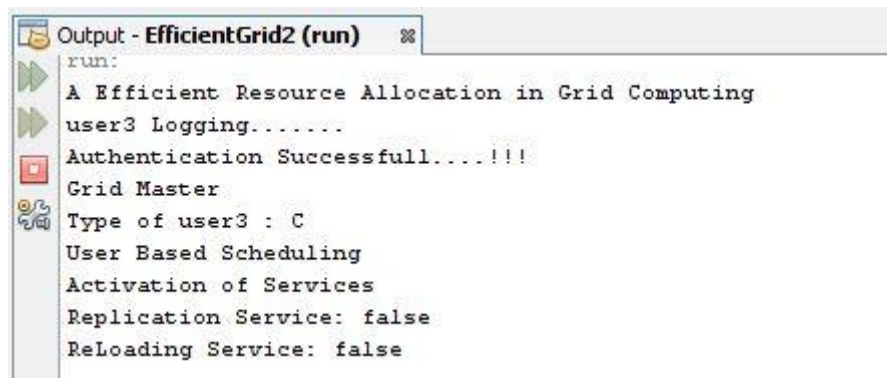
```
A Efficient Resource Allocation in Grid Computing
user2 Logging.....
Authentication Successfull....!!!
Grid Master
Type of user2 : B
User Based Scheduling
Activation of Services
Replication Service: false
ReLoading Service: true
```

**Figure 4.7 Output for Type B after successfully login**

iii. If user is Type C user, the window shown in figure 4.8 will be displayed. No any service will be activated for this category as shown in figure 4.9. This category of users paid no or very less amount to use resources. Therefore we provide minimum share service i.e. 70% success rate. If this percentage will degrade, then we can activate reloading i.e. buffer service. This thing will help to gain 70% success rate by handling VM failure.



**Figure 4.8 Interface for Type C user**



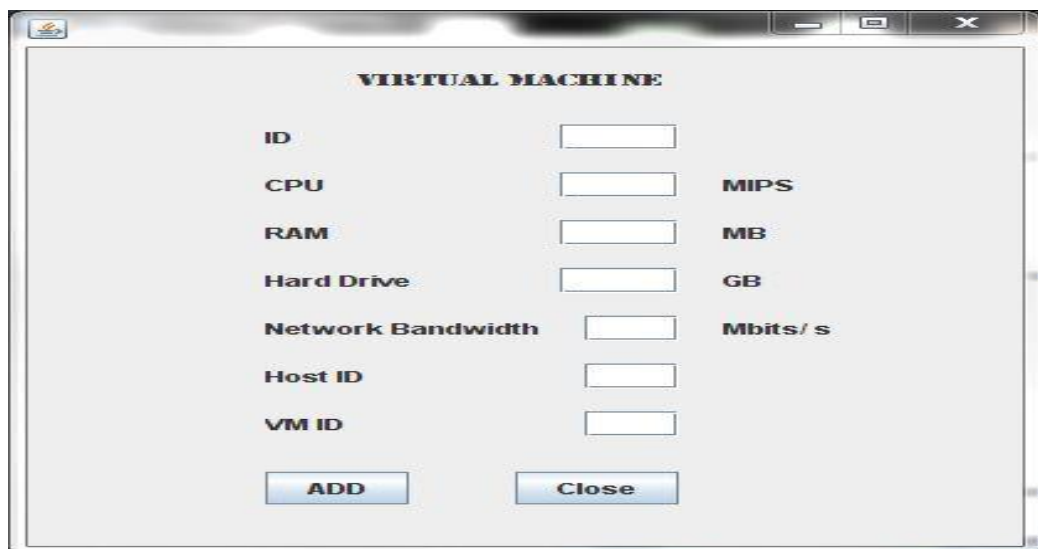
**Figure 4.9 Output for Type C after successfully login**



iv. In this simulation, network failure will not be handled because of security concerns. I assume that user needs to expire the session and reconnect to the grid again in case network failure, because the job or data of each user is very crucial. If user doesn't expire the session, it may be possible that untrusted or third person may resubmit the job(s) of the authorized user on the behalf of him or her. Therefore if there is some network failure, user need to establish new connection to ensure the security.

### 4.2.3 Resource Joining Module

i. After clicking on ADD VM button, the window shown in figure 4.10 will be displayed. The resources can be added to the grid at run time and can adjust into one of the logical cluster (i.e. high end, medium end, or low end resources) according to their configuration (CPU, RAM, Network Bandwidth, and HDD). The output shown in figure 4.11 will be displayed when click on ADD VM button.



**Figure 4.10 Interface for adding resources**

```
Authentication Successfull....!!!  
Grid Master  
Type of user1 : A  
User Based Scheduling  
Activation of Services  
Replication Service: true  
ReLoading Service: true  
Adhoc Resources: Add vm
```

**Figure 4.11 Output after clicking ADD VM button**

ii. Now, we have to enter configuration of resources as shown in figure 4.12

**VIRTUAL MACHINE**

ID	<input type="text" value="41"/>	
CPU	<input type="text" value="100000"/>	<b>MIPS</b>
RAM	<input type="text" value="1024"/>	<b>MB</b>
Hard Drive	<input type="text" value="10"/>	<b>GB</b>
Network Bandwidth	<input type="text" value="200"/>	<b>Mbits/s</b>
Host ID	<input type="text" value="4"/>	
VM ID	<input type="text" value="3"/>	

**Figure 4.12 Interface after adding configuration of resource**

iii. By clicking on ADD button, developed logic will compute mean vectors for each configuration that is CPU, RAM, HD, and Bandwidth as defined bellow:

$$\text{Meanvector\_CPU} = \text{Cpu\_value (e)} / \text{Cpu\_centroid\_value}$$

$$\text{Meanvector\_RAM} = \text{RAM\_value (e)} / \text{RAM\_centroid\_value}$$

$$\text{Meanvector\_HD} = \text{HD\_value (e)} / \text{HD\_centroid\_value}$$

$$\text{MeanVector\_NB} = \text{NB\_value (e)} / \text{NB\_centroid\_value}$$

Where (e) denotes entered values and centroid values of each configuration are constant values taken as central values to be compared with.

iv. By adding all mean vectors, logic will compute the total value which is compared with the threshold values of each cluster.

$$\text{TOTAL} = \text{Meanvector\_CPU} + \text{Meanvector\_RAM} + \text{Meanvector\_HD} + \text{MeanVector\_NB}$$

- v. The cluster having maximum TOTAL value is defined as high cluster, medium TOTAL value is defined as medium cluster, and having low TOTAL value is defined as low cluster.
- vi. As we can see in figure 4.13, there are total of 40 resources already added. When we try to add new resource in grid (as shown in figure 4.12), it will be added into two databases: one where all resources are stored and other in appropriate database according to its configuration.

	A	B	C	D	E	F	G	H	I
21	20	5	4	idle	60000	1024	10	120	
22	21	6	1	idle	60000	2048	20	100	
23	22	6	2	idle	60000	1048	20	100	
24	23	6	3	idle	100000	4096	40	100	
25	24	6	4	idle	20000	512	10	100	
26	25	7	1	idle	100000	4096	20	100	
27	26	7	2	idle	60000	1024	20	100	
28	27	7	3	idle	20000	512	10	100	
29	28	7	4	idle	20000	1024	10	100	
30	29	8	1	idle	200000	2048	20	200	
31	30	8	2	idle	150000	2048	20	200	
32	31	8	3	idle	100000	2048	20	200	
33	32	8	4	idle	40000	512	10	100	
34	33	9	1	idle	120000	2048	20	100	
35	34	9	2	idle	120000	4096	20	100	
36	35	9	3	idle	150000	4096	20	100	
37	36	9	4	idle	10000	2048	10	100	
38	37	10	1	idle	150000	4096	30	100	
39	38	10	2	idle	220000	4096	30	100	
40	39	10	3	idle	20000	1024	10	50	
41	40	10	4	idle	60000	1024	10	120	

**Figure 4.13 Resource.csv file before adding new resource**

- vii. When click on ADD button, the output shown in figure 4.14 will be displayed.

```

Replication Service: true
ReLoading Service: true
Adhoc Resources: Add vm
Total : 5.1819444
Vm has been added succesfully

```

**Figure 4.14 Output after adding resource**

- viii. The resource is added in resource.csv file as shown in figure 4.15. Also as its TOTAL value is 5.18 (shown in figure 4.14), it lies in medium range. So, the resource will also be added into the database where average availability resources are stored as shown in figure 4.16

	A	B	C	D	E	F	G	H	I
21	20	5	4	idle	60000	1024	10	120	
22	21	6	1	idle	60000	2048	20	100	
23	22	6	2	idle	60000	1048	20	100	
24	23	6	3	idle	100000	4096	40	100	
25	24	6	4	idle	20000	512	10	100	
26	25	7	1	idle	100000	4096	20	100	
27	26	7	2	idle	60000	1024	20	100	
28	27	7	3	idle	20000	512	10	100	
29	28	7	4	idle	20000	1024	10	100	
30	29	8	1	idle	200000	2048	20	200	
31	30	8	2	idle	150000	2048	20	200	
32	31	8	3	idle	100000	2048	20	200	
33	32	8	4	idle	40000	512	10	100	
34	33	9	1	idle	120000	2048	20	100	
35	34	9	2	idle	120000	4096	20	100	
36	35	9	3	idle	150000	4096	20	100	
37	36	9	4	idle	10000	2048	10	100	
38	37	10	1	idle	150000	4096	30	100	
39	38	10	2	idle	220000	4096	30	100	
40	39	10	3	idle	20000	1024	10	50	
41	40	10	4	idle	60000	1024	10	120	
42	41	4	3	idle	100000	1024	10	200	
43									

**Figure 4.15 Resource.csv file after adding new resource**

	A	B	C	D	E	F	G	H	I	J	K
1	Id	HostId	VMId	Status	Reliability	CPU	RAM	HD	NB	Value	
2	1	1	1	idle	average	60000	2048	20	100	5.009167	
3	2	1	2	idle	average	60000	1048	20	100	4.032604	
4	6	2	2	idle	average	60000	1024	20	100	4.009167	
5	16	4	4	idle	average	10000	2048	10	100	3.668195	
6	20	5	4	idle	average	60000	1024	10	120	3.709167	
7	21	1	1	idle	average	60000	2048	20	100	5.009167	
8	22	1	2	idle	average	60000	1048	20	100	4.032604	
9	26	2	2	idle	average	60000	1024	20	100	4.009167	
10	36	4	4	idle	average	10000	2048	10	100	3.668195	
11	40	5	4	idle	average	60000	1024	10	120	3.709167	
12	41	4	3	idle	average	100000	1024	10	200	5.181944	
13											

**Figure 4.16 Resourceaverage.csv file after adding new resource**

- ix. In this way we can handle the dynamicity of resources by using this methodology which is used in our simulation.

#### 4.2.4 Resource Allocation Module

In this simulation, choose one of the technique and click on Simulation button. The resources will be allocated to incoming jobs according to the chosen technique. As shown in the following figure 4.17, I choose efficient technique, and then click on Simulation button to allocate jobs according to the proposed technique.

The screenshot displays a web-based interface for resource allocation. It is titled "USER BASED SCHEDULING". Under this title, there are two fields: "Username" with the value "user1" and "User Category" with the value "A". Below these fields is a section titled "ACTIVATION OF SERVICES" which includes two rows: "Replication Service" set to "YES" and "Buffer" set to "YES". To the right of this section is a blue button labeled "ADD VM". At the bottom, there is a section titled "SUBMIT THE JOB" with three radio button options: "FIFO", "AHSWDG", and "Efficient Technique". The "Efficient Technique" option is selected. Below the radio buttons is a blue button labeled "Simulation".

**Figure 4.17 Resource Allocation Interface.**

- i. **FIFO:** - It allocates the first resource in the list to the first job in the job list and then so on. No failure handling in case of job failure will be provided by this technique. Each time resources and users would be treated as same. It may happen that when the job will come from high user, it would be given low configured resource due to which execution time may effect. Similarly for other category of user, sometimes they get benefit sometimes loss in case of performance. It doesn't need to compute anything to find good resource.

- ii. **AHSWDG:** - It first computes the computational capacity of all the resources. Then the resource having maximum computational capacity is allocated to the incoming job. This means it needs to search best resource from all available resources. It doesn't handle failure when job failed during its execution. But it treats all the users equally.
- iii. **Efficient Technique:** - It allocates the resources according to the priority of users. The highly configured resource(s) is allocated to job(s) of user(s) who paid maximum charges. It clusters the resources into three categories to reduce search time for best resource. It also handles the failures to reduce the job failure rate.

### 4.3 Experimental Results 2

To set up the experimental set up for the jobs and to store the information about the clients and resources, I used the following .csv (Comma Separated Values) files. The column values are separated by the comma to store the information.

#### 4.3.1 Client.csv

This file contains the username and password of all the clients as well as SLA (Service Level Agreement) type as shown in figure 4.18

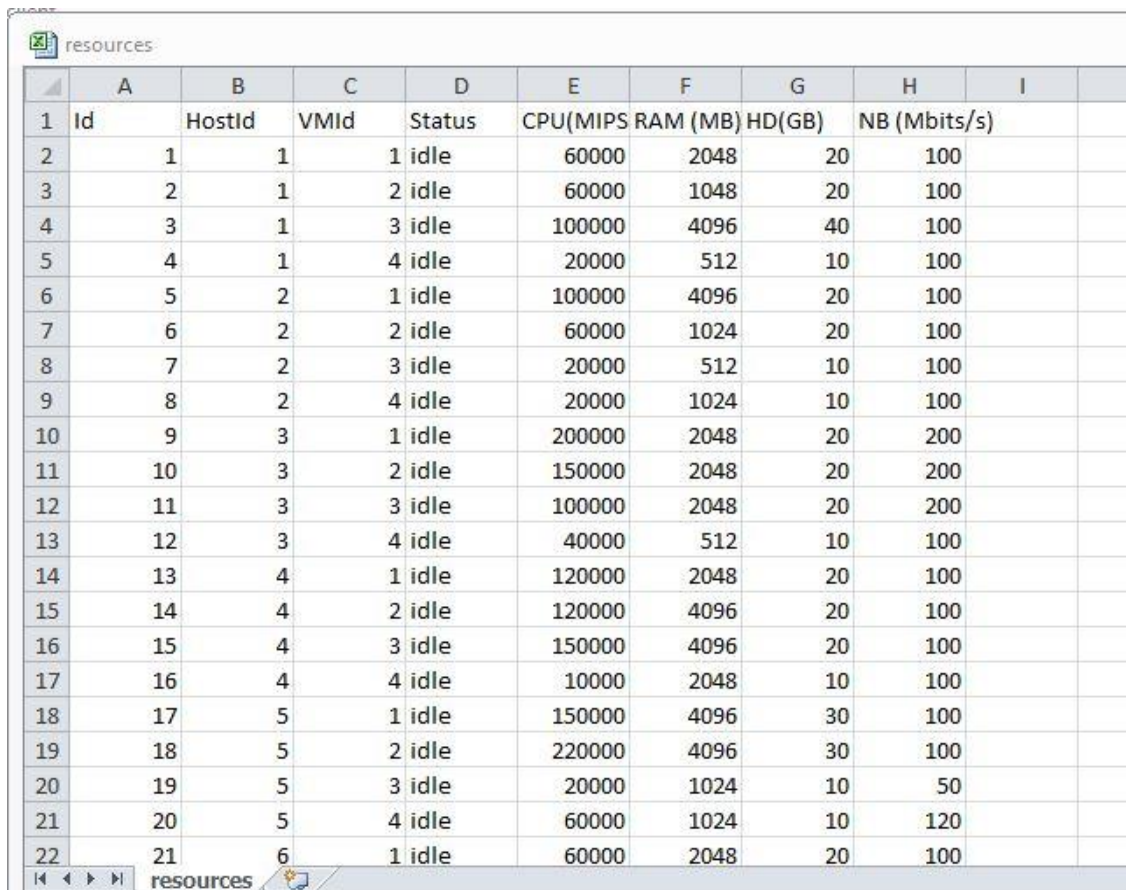
	A	B	C	D	E	F
1	clientID	username	password	SLAtype		
2	101	user1	user1	A		
3	102	user2	user2	B		
4	103	user3	user3	C		
5						
6						
7						
8						
9						
10						

Figure 4.18 Client.csv file



### 4.3.2 Resources.csv

This file contains the information about all available resources. In my simulation, I consider that there are 10 hosts. Each host is having 4 VMs. This means total 40 VMs are there to execute the jobs of the clients. In this file, resources are stored by storing the information regarding id, host id, VM id, status, CPU (in MIPS), RAM (in MB), HD (in GB), and Network Bandwidth i.e. NB (Mbits/s) as shown in figure 4.19



	A	B	C	D	E	F	G	H	I
1	Id	HostId	VMId	Status	CPU(MIPS)	RAM (MB)	HD(GB)	NB (Mbits/s)	
2	1	1	1	idle	60000	2048	20	100	
3	2	1	2	idle	60000	1048	20	100	
4	3	1	3	idle	100000	4096	40	100	
5	4	1	4	idle	20000	512	10	100	
6	5	2	1	idle	100000	4096	20	100	
7	6	2	2	idle	60000	1024	20	100	
8	7	2	3	idle	20000	512	10	100	
9	8	2	4	idle	20000	1024	10	100	
10	9	3	1	idle	200000	2048	20	200	
11	10	3	2	idle	150000	2048	20	200	
12	11	3	3	idle	100000	2048	20	200	
13	12	3	4	idle	40000	512	10	100	
14	13	4	1	idle	120000	2048	20	100	
15	14	4	2	idle	120000	4096	20	100	
16	15	4	3	idle	150000	4096	20	100	
17	16	4	4	idle	10000	2048	10	100	
18	17	5	1	idle	150000	4096	30	100	
19	18	5	2	idle	220000	4096	30	100	
20	19	5	3	idle	20000	1024	10	50	
21	20	5	4	idle	60000	1024	10	120	
22	21	6	1	idle	60000	2048	20	100	

Figure 4.19 Resource.csv file

### 4.3.3 Jobtrace.csv

To simulate the environment and to build the base case scenario, we need the job trace file as shown in figure 4.20. Here 100 jobs will be considered. Each job is having unique id, status (success or fail), and reason of failure. 0 is used for non-occurrence and 1 is used to indicate the occurrence of failure. I am considering that there are three reasons of failure: Host, VM, and Network failure. To set up experimental set up, I consider that normally if user will submit 100 jobs to Grid Data Center, 60 jobs will be executed

successfully and 40 jobs will be failed. 20 jobs will be failed due to VM failure, 12 due to Host failure, and 8 due to Network failure.

	A	B	C	D	E	F	G
1	gridletID	status	VM	Host	Network		
2		1 success	0	0	0		
3		2 fail	1	0	0		
4		3 success	0	0	0		
5		4 success	0	0	0		
6		5 fail	0	1	0		
7		6 fail	1	0	0		
8		7 fail	0	0	1		
9		8 success	0	0	0		
10		9 success	0	0	0		
11		10 success	0	0	0		
12		11 success	0	0	0		
13		12 success	0	0	0		
14		13 success	0	0	0		
15		14 fail	1	0	0		
16		15 fail	0	1	0		
17		16 success	0	0	0		
18		17 fail	1	0	0		
19		18 fail	0	1	0		
20		19 success	0	0	0		
21		20 success	0	0	0		

Figure 4.20 JobTrace.csv file

#### 4.3.4 Resourcehigh.csv

This file contains all the highly configured resources which are having maximum TOTAL value as shown in figure 4.21

	A	B	C	D	E	F	G	H	I	J	K
1	Id	HostId	VMId	Status	Reliability	CPU(in MIPS)	RAM(in MB)	HD(in GB)	NB(in Mbits/s)	Value	
2		3	1	3 idle	high	100000	4096	40	100	8.681944	
3		5	2	1 idle	high	100000	4096	20	100	7.681944	
4		9	3	1 idle	high	200000	2048	20	200	8.363889	
5		10	3	2 idle	high	150000	2048	20	200	7.522917	
6		11	3	3 idle	high	100000	2048	20	200	6.681944	
7		13	4	1 idle	high	120000	2048	20	100	6.018333	
8		14	4	2 idle	high	120000	4096	20	100	8.018333	
9		15	4	3 idle	high	150000	4096	20	100	8.522917	
10		17	5	1 idle	high	150000	4096	30	100	9.022917	
11		18	5	2 idle	high	220000	4096	30	100	10.20028	
12		23	1	3 idle	high	100000	4096	40	100	8.681944	
13		25	2	1 idle	high	100000	4096	20	100	7.681944	
14		29	3	1 idle	high	200000	2048	20	200	8.363889	
15		30	3	2 idle	high	150000	2048	20	200	7.522917	
16		31	3	3 idle	high	100000	2048	20	200	6.681944	
17		33	4	1 idle	high	120000	2048	20	100	6.018333	
18		34	4	2 idle	high	120000	4096	20	100	8.018333	
19		35	4	3 idle	high	150000	4096	20	100	8.522917	
20		37	5	1 idle	high	150000	4096	30	100	9.022917	

Figure 4.21 Resourcehigh.csv file



### 4.3.5 Resourcesaverage.csv

This file contains all the average configured resources which are having medium TOTAL value as shown in figure 4.22

	A	B	C	D	E	F	G	H	I	J	K
1	Id	HostId	VMId	Status	Reliability	CPU(in MIPS)	RAM(in MB)	HD(in GB)	NB(in Mbits/s)	Value	
2	1	1	1	idle	average	60000	2048	20	100	5.009167	
3	2	1	2	idle	average	60000	1048	20	100	4.032604	
4	6	2	2	idle	average	60000	1024	20	100	4.009167	
5	16	4	4	idle	average	10000	2048	10	100	3.668195	
6	20	5	4	idle	average	60000	1024	10	120	3.709167	
7	21	1	1	idle	average	60000	2048	20	100	5.009167	
8	22	1	2	idle	average	60000	1048	20	100	4.032604	
9	26	2	2	idle	average	60000	1024	20	100	4.009167	
10	36	4	4	idle	average	10000	2048	10	100	3.668195	
11	40	5	4	idle	average	60000	1024	10	120	3.709167	
12	41	4	3	idle	average	100000	1024	10	200	5.181944	
13											
14											

Figure 4.22 Resourceaverage.csv file

### 4.3.6 Resourceslow.csv

This file contains all the low configured resources which are having low TOTAL value as shown in figure 4.23

	A	B	C	D	E	F	G	H	I	J	K
1	Id	HostId	VMId	Status	Reliability	CPU(in MIPS)	RAM(in MB)	HD(in GB)	NB(in Mbits/s)	Value	
2	4	1	4	idle	low	20000	512	10	100	2.336389	
3	7	2	3	idle	low	20000	512	10	100	2.336389	
4	19	5	3	idle	low	20000	1024	10	50	2.336389	
5	24	1	4	idle	low	20000	512	10	100	2.336389	
6	27	2	3	idle	low	20000	512	10	100	2.336389	
7	39	5	3	idle	low	20000	1024	10	50	2.336389	
8	12	3	4	idle	low	40000	512	10	100	2.672778	
9	32	3	4	idle	low	40000	512	10	100	2.672778	
10	8	2	4	idle	low	20000	1024	10	100	2.836389	
11	28	2	4	idle	low	20000	1024	10	100	2.836389	
12											
13											

Figure 4.23 Resourcelow.csv file

## 4.4 Experimental Results 4

In this simulation, I compare my proposed technique with two techniques i.e. FIFO and AHSWDG. In this section, I am going to present the outputs of all techniques.

1. **Outputs of FIFO technique:** - According to the base case scenario, there are total 40 resources to allocate to 100 jobs submitted by the user. When the user submits the job, FIFO will allocate the first resource to the first job in the list and so on. It treats all the users and resources in similar way. It doesn't assign any priority. So, resource of any configuration will be allocated to any job. No matter whether the job needs high or low configured resource. It doesn't handle the failure of jobs. As shown in figure 4.24, there are total 40 resources which FIFO can use to run the jobs and all are idle initially. FIFO will use first 13 resources to run 100 jobs submitted by user, as I use threshold of 8 jobs that can be run by 1 VM or resource.

```
Resource Manager
-----
Total Virtual Machines: 40
Total Idle VMs: 40
Total Used VMs: 0

FIFO:First in First OUT Resource Allocation
-----
1 Host no: 1 Vm no. 1
2 Host no: 1 Vm no. 2
3 Host no: 1 Vm no. 3
4 Host no: 1 Vm no. 4
5 Host no: 2 Vm no. 1
6 Host no: 2 Vm no. 2
7 Host no: 2 Vm no. 3
8 Host no: 2 Vm no. 4
9 Host no: 3 Vm no. 1
10 Host no: 3 Vm no. 2
11 Host no: 3 Vm no. 3
12 Host no: 3 Vm no. 4
```

**Figure 4.24 Output 1 for FIFO technique**

As I already discussed that FIFO doesn't handle the failure of job. Therefore when the job will run on resource and if it failed due to any failure i.e. VM, Host, or network, FIFO will give status as fail as shown in figure 4.25. As we can see that

if Fail Component is VM, Host, or Network, FIFO can't handle that failure and give status as fail.

```

Cloud Data Center
-----
Execution as FIFO
-----
CloudletID      Status      Fail Component      ExecutionTime
-----
1               success     null                 148
2               fail        VM                   135
3               success     null                 127
4               success     null                 126
5               fail        Host                  126
6               fail        VM                   125
7               fail        Network               127
8               success     null                 126
9               success     null                 127
10              success     null                 130
11              success     null                 128
12              success     null                 125
13              success     null                 127
14              fail        VM                   126
15              fail        Host                  128
16              success     null                 126
17              fail        VM                   129
18              fail        Host                  140
19              success     null                 127
20              success     null                 124
21              fail        VM                   126

```

**Figure 4.25 Output2 for FIFO**

```

90              success     null                 127
91              success     null                 126
92              fail        VM                   123
93              success     null                 118
94              fail        Network               114
95              success     null                 113
96              fail        VM                   112
97              success     null                 112
98              success     null                 112
99              fail        Host                  112
100             success     null                 113
Number of Resources utilized: 13
Number of Job Success: 60
Number of Job Fail: 40
Execution Time: 12023
Cost: 1906.6666666666667
Search Time: 323362.0

```

**Figure 4.26 Output for Type A user with FIFO technique**

```

98      success      null      112
99      fail         Host      112
100     success      null      111
Number of Resources utilized: 13
Number of Job Success: 60
Number of Job Fail: 40
Execution Time: 13782
Cost: 2183.1333333333337
Search Time: 319994.0

```

**Figure 4.27 Output for Type B user with FIFO technique**

```

96      fail         VM        145
97      success     null      148
98      success     null      146
99      fail         Host      145
100     success     null      147
Number of Resources utilized: 13
Number of Job Success: 60
Number of Job Fail: 40
Execution Time: 14010
Cost: 2221.2666666666667
Search Time: 332505.0

```

**Figure 4.28 Output for Type C user with FIFO technique**

2. **Outputs of AHSWDG technique:** - It also finds the best resource from all the available resources. It computes the computational capacity and then selects the resource with maximum capacity to assign the incoming job(s). Here the thing is that we can benefit the users who need the high computational power which is the drawback of FIFO. But again it is not beneficial to use this technique to allocate the best resource to low category users. This is the drawback of this technique as it also treats all the users at same level. Also it doesn't handle the failures due to which it gives high failure rate than the proposed technique. It computes computational capacity of all the resources each time the job will come and then by sorting, it selects the best resource. Therefore its search time is more than that of FIFO. Again it also needs 13 resources to run 100 jobs of the user (as a base case scenario). It also needs to find best resource each time out of 40 resources according to experimental set up as shown in figure 4.29.

```

Resource Manager
-----
Total Virtual Machines: 40
Total Idle VMs: 40
Total Used VMs: 0

AHSWDG: An Ant Based Heuristic Approach to Scheduling & Workload Distribution in
-----

```

**Figure 4.29 Output 1 for AHSWDG technique**

As it doesn't handle the failure of the jobs, it also gives the status as fail when job failed due to VM, Host, or Network failure as shown in figure 4.30

```

Cloud Data Center
-----
Execution as major AHSWDG Model
-----

```

CloudletID	Status	Fail Component	ExecutionTime
1	success	null	143
2	fail	VM	139
3	success	null	122
4	success	null	123
5	fail	Host	124
6	fail	VM	126
7	fail	Network	128
8	success	null	120
9	success	null	118
10	success	null	119
11	success	null	114
12	success	null	110
13	success	null	101
14	fail	VM	101
15	fail	Host	101
16	success	null	101
17	fail	VM	100
18	fail	Host	97
19	success	null	92
20	success	null	93

**Figure 4.30 Output 2 for AHSWDG technique**

```

94      fail      Network      63
95      success   null        64
96      fail      VM          63
97      success   null        62
98      success   null        63
99      fail      Host        63
100     success   null        63

Number of Resources utilized: 13
Number of Job Success: 60
Number of Job Fail: 40
Execution Time: 7812
Cost: 1239.3333333333335
Search Time: 2170185.0

```

**Figure 4.31 Output for Type A user with AHSWDG technique**

```

--          ----          -----          --
95          success          null          61
96          fail            VM            60
97          success          null          60
98          success          null          61
99          fail            Host          60
100         success          null          60
Number of Resources utilized: 13
Number of Job Success: 60
Number of Job Fail: 40
Execution Time: 8450
Cost: 1334.6666666666667
Search Time: 2813541.0

```

**Figure 4.32 Output for Type B user with AHSWDG technique**

```

--          -----          -----          --
98          success          null          84
99          fail            Host          84
100         success          null          84
Number of Resources utilized: 13
Number of Job Success: 60
Number of Job Fail: 40
Execution Time: 13032
Cost: 2068.7333333333336
Search Time: 2319836.0

```

**Figure 4.33 Output for Type C user with AHSWDG technique**

3. **Efficient Technique:** - It allocates the resources according to the SLA type of user. The highly configured resource(s) to highly paid user(s). Similarly medium configured to medium paid and low configured to the low paid users. In this way, when the request comes from any type of user, it searches resource only from appropriate cluster. In this way, it doesn't need to search fittest resource from all available resources. Hence it reduces the search time as compared to AHSWDG technique. It also handles the failure when job failed due to VM or Host. But this activation is also done on the basis of the user type, so that no extra resources will be wasted for the medium or low end users. In this way, job failure rate will be decreased as compare to FIFO and AHSWDG. As a base case scenario, to run 100 jobs of highly paid user, algorithm need to search from 20 resources as shown in figure 4.34 which are stored in resourcehigh.csv file as shown in figure 4.21.



```

Resource Manager
-----
Total Virtual Machines: 20
Total Idle VMs: 20
Total Used VMs: 0

Replication Management
-----
3 Host no: 1 Vm no. 3
5 Host no: 2 Vm no. 1
9 Host no: 3 Vm no. 1
10 Host no: 3 Vm no. 2
11 Host no: 3 Vm no. 3
13 Host no: 4 Vm no. 1
14 Host no: 4 Vm no. 2
15 Host no: 4 Vm no. 3
17 Host no: 5 Vm no. 1
18 Host no: 5 Vm no. 2
23 Host no: 1 Vm no. 3
25 Host no: 2 Vm no. 1
29 Host no: 3 Vm no. 1
30 Host no: 3 Vm no. 2
31 Host no: 3 Vm no. 3
33 Host no: 4 Vm no. 1
34 Host no: 4 Vm no. 2
35 Host no: 4 Vm no. 3
37 Host no: 5 Vm no. 1
38 Host no: 5 Vm no. 2

```

**Figure 4.34 Output 1 for EFFICIENT technique**

As we can see in figure 4.35, when job failed due to VM or Host failure for type A user, it can be handled by replicating it on other resource on same host or other host. It gives the status as success because it handles the failure successfully. In this way it gives more success rate. It first sorts list of the appropriate resources, then choose the first best resource to allocate to incoming job.

Also to balance the usage of one machine, algorithm will decrease the total value according to which resources are sorted by 1. Whenever resource is allocated to the incoming job, its total value is decreased by 1 due to which it moves down in this list. The lower resources moves upward in the sorted list and upper resources moves down in the list as their values are decreased.

Execution as Efficient Resource Allocation Technique			
CloudletID	Status	Fail Component	ExecutionTime
1	success	null	137
2	success	VM	127
3	success	null	126
4	success	null	127
5	success	Host	143
6	success	VM	144
7	fail	Network	140
8	success	null	142
9	success	null	142
10	success	null	153
11	success	null	143
12	success	null	144
13	success	null	143
14	success	VM	141
15	success	Host	143
16	success	null	142
17	success	VM	143
18	success	Host	142
19	success	null	143
20	success	null	144
21	success	VM	137
22	success	null	126
23	fail	Network	140
24	success	null	133
25	success	null	120
26	success	VM	116

Figure 4.35 Output 2 for EFFICIENT technique

```

96      success      VM      60
97      success      null     61
98      success      null     60
99      success      Host    61
100     success      null     60
Number of Resources utilized: 25
Number of Job Success: 92
Number of Job Fail: 8
Execution Time: 9055
Cost: 2625.0
Search Time: 1396425.0

```

Figure 4.36 Output for Type A user with EFFICIENT technique

```

99      fail      Host    65
100     success   null    62
Number of Resources utilized: 13
Number of Job Success: 80
Number of Job Fail: 20
Execution Time: 10247
Cost: 1620.6666666666667
Search Time: 984522.0

```

Figure 4.37 Output for Type B user with EFFICIENT technique



```

94          fail          Network          81
95          success       null          81
96          fail          VM            81
97          success       null          81
98          success       null          81
99          fail          Host         80
100         success       null          81
Number of Resources utilized: 13
Number of Job Success: 70
Number of Job Fail: 30
Execution Time: 11295
Cost: 977.6
Search Time: 1225601.0

```

**Figure 4.38 Output for Type C user with EFFICIENT technique**

## 4.5 Experimental Results 5

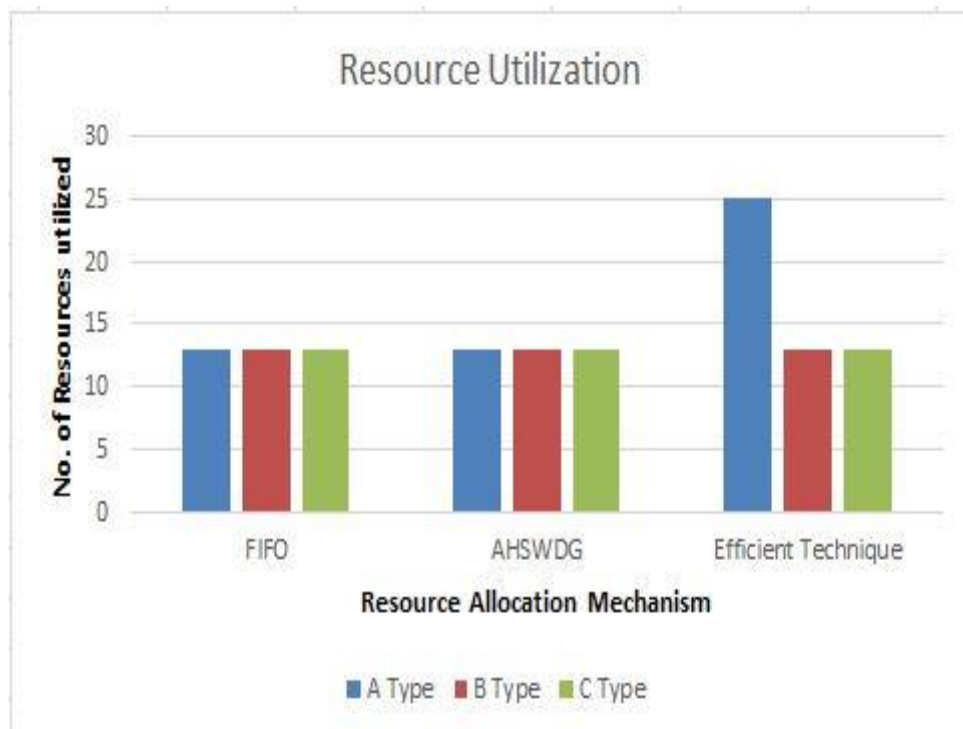
In this section, the results are shown with the help of graphs. The comparisons are done on the basis of SLA type of users and various resource allocation techniques. The proposed technique aims to reduce the failure rate, search time as compared to AHSWDG, increase success rate, to optimize the results on the basis of execution time and cost.

### 4.5.1 Resource Utilization Comparison

Comparison on the basis of total number of resource used as per base case scenario and threshold assumed is shown in the following bar graph (shown in figure 4.39). As I assumed that one VM can run 8 jobs. To run 100 jobs of a user, all the technique need 13 resources except user A for proposed technique. This is because, for Type A user algorithm will activate Replication Service due to which one job will run on two resources. The comparison is shown in table 4.1.

**Table 4.1 Comparison Table of Resource Utilization**

Resource Utilization	FIFO	AHSWDG	Efficient Technique
A Type	13	13	25
B Type	13	13	13
C Type	13	13	13



**Figure 4.39 Comparison graph of Resource Utilization**

#### 4.5.2 Job Success Comparison

Comparison on the basis of number of job executed successfully is shown in the following bar graph (shown in figure 4.40). This comparison is shown in table 4.2. It is noticed that by providing fault tolerance in the proposed technique, number of successful jobs is more than both the techniques. FIFO and AHSWDG don't provide fault tolerance. Failed jobs need to resubmit in both techniques. As per base case scenario, it is assumed that when user will submit 100 jobs to run 60 jobs will run successfully, 40 will fail. Out of 40, 20 will fail due to VM failure, 12 due to Host failure, and 8 due to Network failure. Therefore, proposed technique will handle the VM failure for Type B user, VM and Host failure for Type A user as already discussed in previous section. Also it provides 70 % success rate to Type C users. Therefore it gives high success rate than other two techniques which do not provide fault tolerance.

**Table 4.2 Comparison Table of Job Success**

Job Success	FIFO	AHSWDG	Efficient Technique
A Type	60	60	92
B Type	60	60	80
C Type	60	60	70



**Figure 4.40 Comparison graph of Job Success**

### 4.5.3 Job Failure Comparison

Comparison on the basis of number of job failed during execution is shown in the following bar chart (shown in Figure 4.41). This comparison is shown in table 4.3. It is noticed that number of jobs failed in proposed technique is less than that of FIFO and AHSWDG. The reason for this is fault tolerance provided by the proposed technique.

**Table 4.3 Comparison Table of Job failure**

Job Failure	FIFO	AHSWDG	Efficient Technique
A Type	40	40	8
B Type	40	40	20
C Type	40	40	30



**Figure 4.41 Comparison graph of Job Failure**

#### 4.5.4 Search Time Comparison

Comparison on the basis of search time is shown in the following bar chart (shown in Figure 4.42). This comparison is shown in table 4.4. In simulation, I consider search time in nanoseconds. It is noticed that proposed technique takes less time as compared to AHSWDG technique. Because proposed technique takes into the consideration both the SLA type of the user and the resource availability (i.e. high, medium or low). It needs to search from fewer resources as compared to AHSWDG.

Search time also considers computational time of techniques. For example: To search fittest resource, AHSWDG needs to compute the computational capacity for all the resources each time the new job will come and also then it performs sorting to allocate the resource having the maximum computational capacity.

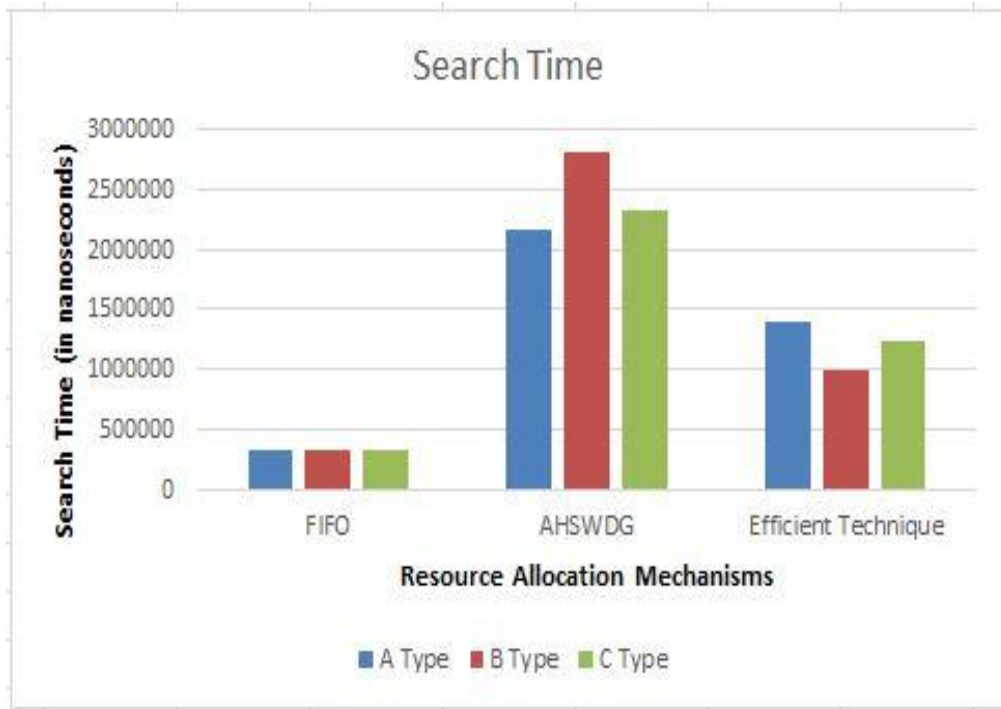
The proposed technique only needs to sort the resources of the appropriate cluster having less number of resources as compared to the all available resources in grid. Then it allocates the best resource at the top to incoming job. Therefore its search time is less than AHSWDG.

As it is also noticed that FIFO takes less search time than both techniques, this is because it allocate the first resource in the queue to first incoming jobs and so on. No any computations need to be done to choose best resource. But it is not beneficial to use FIFO technique because it may be possible that it allocates low configured resource to job which require high configured resource. It may also be possible that it allocates very high configured resource to job which doesn't require that much capacity or configuration.

The AHSWDG is better than FIFO in this aspect, because it allocates the resource with maximum computational capacity to incoming job(s). If the job which requires high computations will always assigns to high configured resource. This is the drawback of FIFO as discussed above. But in case of job which requires low configured resource, AHSWDG will allocate again high configured resource to this job. The proposed technique tackles all these problems in addition to fault tolerance.

**Table 4.4 Comparison Table of Search Time (in nanoseconds)**

Search Time	FIFO	AHSWDG	Efficient Technique
A Type	323362	2170185	1396425
B Type	319994	2813541	984522
C Type	332505	2319836	1225601



**Figure 4.42 Comparison graph of Search Time**

#### 4.5.5 Execution Time Comparison

Comparison on the basis of execution time is shown in the following bar chart (shown in Figure 4.43). This comparison is shown in table 4.5. The aim of the proposed technique is to optimize the performance of the grid not to reduce or to increase the performance. In case of replication service to recover the failure, it needs to execute the job on more than one resource, which may lead to the more execution time than the other technique. So the proposed technique provides the optimized results. According to the SLA agreement type and services need to provide to the users to benefit the users accordingly, the proposed technique gives the optimized results.

As already discussed that FIFO may allocates low configured resources to jobs of Type A user because it never differentiate between users and resources. AHSWDG is better than FIFO in most cases.

**Table 4.5 Comparison Table of Execution Time (in milliseconds)**

Execution Time	FIFO	AHSWDG	Efficient Technique
A Type	12023	7812	9055
B Type	13782	8450	10247
C Type	14010	13032	11295



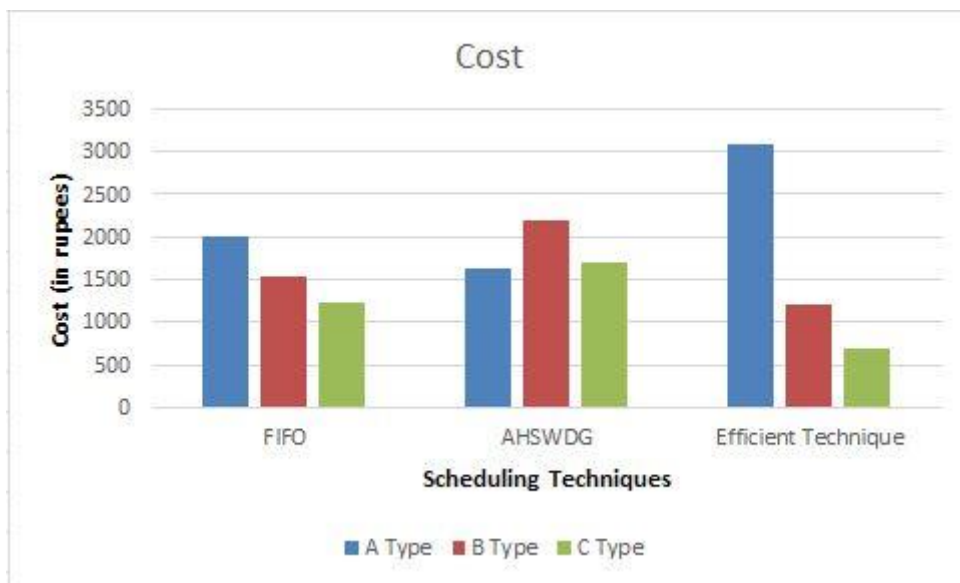
**Figure 4.43 Comparison graph of Execution Time**

#### 4.5.6 Cost Comparison

Comparison on the basis of total cost is shown in the following bar chart (shown in Figure 4.44). This comparison is shown in table 4.6. As I already discussed that aim of the proposed technique is not to reduce or to increase the cost, the aim is to optimize it. As it is noticed that for type A user, cost of proposed technique is more than that of others. It is due to the usage of multiple resources when providing the replication service only to this category of users.

**Table 4.6 Comparison Table of Cost (in Rs)**

Cost	FIFO	AHSWDG	Efficient Technique
A Type	1906.66	1239.33	2625
B Type	2183.13	1334.66	1620.66
C Type	2221.66	2068.73	977.6



**Figure 4.44 Comparison graph of Cost**



### **CONCLUSION AND FUTURE WORK**

---

Resource Allocation is the process of allocating the resources to the tasks in efficient manner so that throughput of the system will increase and user requirements will be satisfy. In static scheduling, state of all system resources and tasks are known in advance. These states can't be changed. This limits such type of schedulers to specific problems and systems. Therefore I conclude that dynamic scheduling will give us better solutions in dynamic environment. As we know that grid environment is dynamic in nature, so handling the dynamic resources is the big challenge for researchers. The proposed mechanism can handle these resources in better way. The mechanism will be used to allocate the resource(s) to the incoming job(s) efficiently by reducing the search time.

The algorithm will be based on user based scheduling instead of job based scheduling. So by using this technique, we can prioritize the users on the basis of their SLA (Service Level Agreement) type. If the user is of high priority, highly configured resources would be allocated to assign his/her jobs. Similarly, medium configured for medium and low configured resources will be used for low priority users.

The algorithm also provides the capability of handling the failures which can't be handled in existing technique such as AHSWDG. In this way, the proposed algorithm gives the high success rate of jobs by handling VM and Host failure. In this way, the reliability can also be achieved.

The future plan is to extend the work so as to provide more reliable and efficient performance in case of network failure also.

## LIST OF REFERENCES

---

### I. Books

Joshy Joseph, Craig Fellenstein, “Grid Computing”, Published by Prentice Hall Professional, 2004

### II. Research Papers

- [1] Albert Y. Zomaya, Cheng-Zhong Xu, Hameed Hussain, Ilias Rentifis, Muhammad Bilal Qureshi, Muhammad Shuaib Qureshi, Maryam Mehri Dehnavi, Nasro Min-Allah, Nikos Tziritas, Samee U. Khan, Thanasis Loukopoulos (2014), “*Survey on Grid Resource Allocation Mechanisms*”, Springer Science+Business Media Dordrecht , pp. 399-441 .
- [2] Aboamama Atahar Ahmed, Abdul Hanan Abdullah, Adil Yousif, Mohammed Bakri Bashir, Muhammad Shafie Bin Abd Latiff, Yahaya Coulibaly (2011), “*A HYBRID RESOURCE DISCOVERY MODEL FOR GRID COMPUTING*”, International Journal of Grid Computing & Applications (IJGCA) Vol.2, No.3.
- [3] Esmat Rashedi, Hossein Nezamabadi-pour, Saeid Saryazdi (2009), “*GSA: A Gravitational Search Algorithm*”, Springer Science+.
- [4] Amirreza Zarrabi, Khairulmizam Samsudin (2013), “*Task scheduling on computational Grids using Gravitational Search Algorithm*”, Springer Science+Business Media New York, pp. 1001-1011
- [5] Belabbas Yagoubi, Meriem Meddeber (2011), “*Towards Hybrid Dependent Tasks Assignment for Grid Computing*”,
- [6] Birje M.N, Manvi S.S, (2005), “*An agent based resource allocation on model for grid computing*”, Services Computing, 2005 IEEE International Conference on (Volume: 1), pp. 311-314
- [7] K. Somasundaram, S. Radhakrishnan, (2009), “*Task Resource Allocation in Grid using Swift Scheduler*” Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. IV , pp. 158-166

- [8] D.Maruthanayagam, Dr. R.Uma Rani, (2011),”*Improved Ant Colony Optimization for Grid Scheduling*” IJCSET, Vol. 1, Issue 10, pp. 596-604
- [9] Rose Al Qasim, Taisir Eldos, (2013), “*On The Performance of the Gravitational Search Algorithm*”, International Journal of Advanced Computer Science and Applications (IJACSA), Vol.4, No. 8
- [10] H.H. Hoos, T. Stutzle, (2000) “*Max-Min Ant System*”, Future Generation Computer Systems, pp. 889-914
- [11] Belabbas Yagoubi, Meriem Meddeber, Walid Kadri, (2011),”*A Static Tasks Assignment for Grid Computing*”, IJACSA, Vol.4
- [12] Hui Yan, Ming- Hui Wu, Xing Li, Xue-Qin, (2005), “An Improved Ant Algorithm for Job Scheduling in Grid Computing”, International Conference on Machine Learning and Cybernetics, Volume 5, pp. 2957-2961.
- [13] Bhavik Kothari, Chittaranjan Hota, Sunita Bansal, (2011),” *Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm*”, International Journals of Computer Science Issues, Vol.8, Issue 2
- [14] Chun-Yan LIU, Cheng-Ming ZOU, Pei WU, (2014), “*A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing*”, 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science.
- [15] C. Blum, M. Dorigo, (2005), “*Ant Colony Optimization Theory: A Survey*” Theoretical Computer Science, Volume 344, Issue 2-3, pp. 243-278.
- [16] A. Rama Mohan Reddy, K. Sathish, (2013), “*Maximizing Computational Profit in Grid Resource Allocation using Dynamic Algorithm*”, Global Journals of Computer Science and Technology, Volume 13, Issue 2, Version 1.0
- [17] Husna Jurnal Abdul Nasir, Ku. Mahamud, Ku. Ruhana, (2010), “*Ant Colony Algorithm for Job Scheduling in Grid Computing*”, Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, pp. 40-44.

- [18] Ankur Kumar, Anuj Kumar, Rohit Saxena, Shailesh Saxena, (2015) “*AHSWDG: An Ant Based Heuristic Approach to Scheduling & Workload Distribution in Computational Grids*”, IEEE International Conference on Computational Intelligence & Communication Technology, pp. 569-574.
- [19] Babita Pandey, Harsh Bansal, Kewal Krishan, (2012) “*Intelligent Methods for Resource Allocation in Grid Computing*”, International Journals of Computer Applications (IJCA), Volume 47-No. 6.
- [20] Abdul Hanan Abdullah, Chai Chompoo-inwai, Mohd Noor Sap, Siriluck Lorpunmanee, (2007) “*An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment*”, International Journals of Computer and Information Engineering, pp. 469-476

### **III. Websites**

[en.m.wikipedia.org/wiki/Grid\\_computing](http://en.m.wikipedia.org/wiki/Grid_computing)

[www.gridcomputing.com](http://www.gridcomputing.com)

[www.redbooks.com](http://www.redbooks.com)